НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра автоматики та управління в технічних системах

(повна назва кафедри)

«На правах рукопису»
УДК _____

«До захисту допущено»

Завідувач кафедри

_____ _____
(підпис) (ініціали, прізвище)

"___"_____20__ р.

Магістерська дисертація

зі спеціальності (спеціалізації) 126 Інформаційні системи та технології
(код і назва спеціальності)

на тему: «Система взаємодії з користувачем на основі чат-ботів»

Виконав (-ла): студент (-ка) 6(2) курсу, групи ІА-з82мп
(шифр групи)

Моісеєнко Петро Юрійович
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник доц.каф. АУТС, к.т.н Букасов М.М.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____
(назва розділу) (науковий ступінь, вчене звання,, прізвище, ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет (інститут) інформатики та обчислювальної техніки
(повна назва)

Кафедра автоматики та управління в технічних системах
(повна назва)

Рівень вищої освіти – другий (магістерський)

Спеціальність (спеціалізація) 126 – Інформаційні системи та технології
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ __О.І. Ролік_____
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Моісеєнку Петру Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації «Система взаємодії з користувачем на основі чат-ботів»
науковий керівник дисертації Букасов Максим Михайлович, к.т.н,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження: полегшення інтеграції CRM-системи з чат-ботом.

4. Предмет дослідження: засоби миттєвого обміну повідомленнями.

5. Перелік завдань, які потрібно розробити: розробити чат-бота, розробити CRM-систему, описати сценарії використання інтегрованих системи чат-боту та CRM-системи, спроектувати архітектуру та реалізувати рішення.

6. Орієнтовний перелік ілюстративного (графічного) матеріалу: Use-case діаграми, структурна схема системи, схема бази даних, інтерфейс системи.

7. Орієнтовний перелік публікацій _____

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Отримання та узгодження вхідних даних	05.09.2019 – 20.09.2019	
2	Вивчення об’єкту дослідження	21.09.2019 – 30.09.2019	
3	Розробка програмної моделі	01.10.2019 – 14.10.2019	
4	Реалізації системи	15.10.2019 – 20.11.2019	
5	Проектування документації	21.11.2019 – 28.11.2019	
6	Оформлення документації	29.11.2019 – 02.12.2019	
7	Подання роботи до попереднього захисту	03.12.2019	

Студент _____ П.Ю.Моісеєнко
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ М.М. Букасов
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Моісеєнко П.Ю. Система взаємодії з користувачем на основі чат-ботів. КПІ ім. Ігоря Сикорського, Київ, 2019. Магістерську дисертацію виконано на 104 сторінках, що містить 12 розділів, 39 ілюстрацій, 29 таблиць, 20 джерел в переліку посилань та 8 додатків.

Актуальність: Для автоматизації взаємодії з користувачами та автоматизації бізнес-процесів компанії використовують CRM-системи. Реалізована система дозволяє інтегрувати серверну частину існуючої CRM-системи з месенджером Telegram та за допомогою чат-боту дозволяє користувачу самостійно керувати життєвим циклом послуг. Така автоматизація дозволяє компанії бути ближчим до споживача без великих витрат.

Мета: Мета даної роботи це рішення для зручного створення чат-ботів в месенджері Telegram та демонстрація роботи чат-боту на прикладі CRM-системи телекомунікаційної компанії. Рішення має управляти бізнес-процесами заданими налаштуванням за рахунок інтеграції з CRM-системою.

Задачі: Для досягнення мети були поставлені та розв’язані такі завдання:

1) встановлення технології на яких буде рішення;
2) розроблено чат-бота;
3) розроблено серверну частину CRM-системи;
4) інтегровано дві системи.

Об’єкт: об'єктом дослідження в роботі є полегшення інтеграції CRM-системи з чат-ботом.

Предмет: предметом дослідження є засоби миттєвого обміну повідомленнями.

В першому розділі розглянуто актуальність проблеми створення чат-боту інтегрованого з CRM-системою.

В другому розділі подано сформовано список функціональних та нефункціональних вимог до системи.

В третьому розділі подано обґрунтування обраних технології для розробки.

В четвертому-десятому розділах описані такі нюанси розробленої системи як: структурна схема системи, можливі сценарії використання, опис ER-моделі та бізнес-логіки. Також надано приклад файлу налаштувань для системи з поясненнями.

В одинадцятому розділі описано тестування системи та порівняно поставлені функціональні вимоги та прецеденти системи.

В дванадцятому розділі описана розробка стартап проекту на основі виконаної магістерської дисертації.

Ключові слова: АВТОМАТИЗАЦІЯ, CRM-СИСТЕМА, ІНТЕГРАЦІЯ, БОТ, ЧАТ-БОТ, МЕСЕНДЖЕР.

ABSTRACT

Moiseienko P.Y. Customer interaction system based on chatbots. Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, 2019. The master's thesis is made on 104 pages, containing, 12 sections, 39 illustrations, 29 tables, 20 sources in the list of links and 8 additions.

Relevance: To manage customer relationships and manage business processes companies use CRM systems. The system implemented in this work allows a customer to integrate a server part of an existing CRM system with a Telegram messanger. The system based on chatbot allows the customer to manage independently the service life cycle. Such automation allows the company to be closer to the customer without great costs.

Purpose: The purpose of this work is a solution for easy creation of chatbots in the Telegram messanger and demonstration on an example of a telecommunication company's CRM system. The solution has to manage the set upped business processes by integration with the CRM system.

The tasks: To achieve this goal, the following tasks have been set and solved:

1) establish technologies on which the decision will be made;
2) implement the chatbot;
3) implement server side of the CRM system
4) integrate CRM system with chatbot.

Object: The object of the research is to facilitate the integration of a CRM system with a chatbot.

Subject matter: The subject of the research is instant messaging systems.

The first section provides a topicality of creating chatbot integrated with CRM system.

The second section describes a list of functional and non-functional requirements for the system.

The third section provides a justification for the selected technologies for development process.

The sections within forth and tenth describe the nuances of the developed system such as: a system structure, a possible use cases, description of a ER-model and a business logic. An example of configuration file for the system with explanations is also provided.

The eleventh section describes system testing and comparatively functional requirements and system precendents.

The twelfth section describes the development of a startup project based on the a master’s thesis.

Keywords: AUTOMATION, CRM-SYSTEM, INTEGRATION, BOT, CHATBOT, MESSANGER.

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ 9
ВСТУП 10
1 АКТУАЛЬНІСТЬ ПРОБЛЕМИ 12
1.1 Історія розвитку засобів обміну миттєвими повідомленнями 12
1.2 Застосування чат-ботів для бізнесу 15
2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ 17
2.1 Формування функціональних вимог 17
2.2 Формування нефункціональні вимог 18
3 ВИБІР ТА ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ 19
4 СТРУКТУРНА СХЕМА СИСТЕМИ 23
5 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ 25
6 РЕАЛІЗАЦІЯ ЕЛЕМЕНТІВ СИСТЕМИ 33
6.1 Реалізація елементів CRM-системи 33
6.2 Реалізація елементів системи чат-боту 37
7 ER-МОДЕЛЬ 42
8 РЕАЛІЗАЦІЯ БІЗНЕС-ЛОГІКИ КОРИСТУВАЧА 50
9 НАЛАШТУВАННЯ ЧАТ-БОТУ 55
10 ІНТЕРФЕЙС КОРИСТУВАЧА 59
11 ТЕСТУВАННЯ СИСТЕМИ 66
12 СТАРТАП-ПРОЕКТ 80
ВИСНОВКИ 99
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 103
Додаток А **Ошибка! Закладка не определена.**
Додаток Б **Ошибка! Закладка не определена.**
Додаток В **Ошибка! Закладка не определена.**
Додаток Г **Ошибка! Закладка не определена.**
Додаток Д **Ошибка! Закладка не определена.**
Додаток Ж **Ошибка! Закладка не определена.**
Додаток К **Ошибка! Закладка не определена.**
Додаток Л **Ошибка! Закладка не определена.**

ПЕРЕЛІК ТЕРМІНІВ

API – прикладний програмний інтерфейс – набір методів для взаємодії різних компонентів системи

CI/CD – практика розробки програмного забезпечення

CRM – система управляння взаємодії з клієнтами

CRUD – функції управління даними: створення, зчитування, зміна та видалення відповідно

ER-модель – модель «сутність-зв'язок» – це засіб опису моделей даних

HTTP – протокол передачі даних

InlineKeyboardButton – кнопка, яка розташовується під повідомлення в Telegram

InlineKeyboardMarkup – набір кнопок InlineKeyboardButton, які розташовуються під повідомленням в Telegram

JPA – інтерфейс для фреймворків, які зв'язують сутності бази даних та Java об'єкти

MVC – архітектурний шаблон розробки

QA – забезпечення якості, відповідно QA-інженер – спеціаліст з забезпечення якості

REST – підхід до архітектури мережевих протоколів

UML – уніфікована мова моделювання, призначена для графічного створення абстрактної моделі системи

ВСТУП

Сучасне суспільство поступово розвивається і з появою комп’ютерів та інтернету почали з’являються інтернет-сайти. В наш час важко уявити людину, яка не користується будь-якими інтернет-сервісами. Це призводить до того, що взаємодія з кінцевими користувачами автоматизується і для цього слугує CRM-система.

CRM – це поняття, яке охоплює концепції, які використовуються компаніями для автоматизації взаємодії з користувачами. Внаслідок реалізації CRM компанії збирають, зберігають та аналізують інформацію про своїх споживачів, партнерів.

З точки зору користувача управління власними продуктами стало легшим з появою CRM-систем адже відтепер для процесу управління життєвим циклом послуг був потрібен лише комп’ютер та інтернет. Проте час не стоїть на місці і з появою смартфонів та планшетів почали виникати месенджери. І хоча причиною їх виникнення була базова потреба людини у спілкуванні, але і цей ринок поступово розвивався. На сьогодні месенджер слугує не лише для зв’язку між людьми, а виконує безліч інших функцій. З появою каналів тепер кожен в ньому може вести свій блог, отримувати новини, грати в ігри. Для автоматизації почали використовуватись чат-боти.

Чат-бот це комп’ютерна програма, яка веде діалог за допомогою слухових чи текстових методів. Часто його використовують для надання необхідної інформації – вони існують для досягнення практичних цілей таких як обслуговування клієнтів.

Майже кожна людина у повсякденному житті використовує смартфон, і більшість користується месенджерами. Для багатьох це є центром отримання інформації. Саме тому, з метою спрощення життя користувачами, компанії використовують чат-ботів. Для інформування на зміну СМС повідомленням приходять месенджери.

Сфера, основна мета якої полягає в спілкуванні та отриманню інформації є телекомунікаційна. І якщо раніше керувати послугами було зручно за допомогою СМС повідомлень та дзвінкам, то зараз доцільно використовувати для цих цілей месенджери. Оскільки всі телекомунікаційні компанії мають власні CRM-системи для автоматизації взаємодії з користувачем, то чат-бот може використовувати існуюче серверне API для своєї роботи. Проте тут існує проблема – при додаванні нових функції до сервісу чи його оновленні програму для чат-боту теж доведеться оновлювати.

Мета даної роботи це рішення для автоматизації взаємодії з користувачем за допомогою створення чат-боту в месенджері Telegram з інтеграцією з існуючим серверним рішенням для CRM-системи. Для цього буде написана програма, за допомогою якої можна створювати бота, лише задав йому відповідні налаштування. Для демонстрації цього, окрім «конструктора» для бота, буде реалізована серверна частина призначена для імітування CRM-системи у сфері телекомунікацій.

Мета продукту це рішення для швидкого отримання клієнтом інформації щодо підключених послуг, можливість підключення нових продуктів або відключення існуючих. Також клієнт буде інформуватися о нових продуктах чи послугах компанії та отримувати системні повідомлення.

Рішення має управляти бізнес-процесами, тобто автоматизувати послідовні операції, які виконуються співробітниками організації, тобто формалізувати схему взаємодії з клієнтами. Як наслідок це зниження операційних витрат менеджерів.

Завдяки автоматизації системи, клієнт зможе підписатись на повідомлення від бота, авторизуватись за допомогою номеру телефону та матиме можливість переглянути свої особисті дані, інформацію за своїм рахунком, наявні сервіси, історію використання бонусів.

1 АКТУАЛЬНІСТЬ ПРОБЛЕМИ

1.1 Історія розвитку засобів обміну миттєвими повідомленнями

Люди є соціальними істотами, а отже спілкування це одна з базових потреб людства. Це є поштовхом до еволюції, адже саме завдячуючи цьому ми маємо змогу розвиватися, оскільки виникає постійний процес обміну досвідом та знаннями. Але після відходу від сімейно-родинного устрою виникла необхідність в дистанційному спілкуванні.

Спочатку були листи, потім з’явився телеграфний зв'язок, ще згодом – телефонний. Поступово проміжок часу між відправленням повідомлення до його отримання скорочувався. З появою комп’ютерів та інтернету з’явилась електронна пошта. Проте в цих засобах обміну повідомлень був суттєвий недолік – в спілкуванні могли брати участь лише дві людини. Саме тому почали виникли чати.

Першим онлайн чатом був Talkomatic створений Дугом Брауном та Девідом Р. Уоллі у 1973 році в системі PLATO (Programmed Logic for Automated Teaching Operations), яка була першою системою для електронного навчання та була створена в університету штату Іллінойс. Він був призначений для спілкування невеликою групою людей одночасно. Цікавою особливістю було те, що кожен учасник мав власну частину екрану, а повідомлення відображались у реальному часі буква за буквою як він їх друкував на клавіатурі.

З часом, комп’ютери ставали більш доступними і все більше людей хотіли спілкування. Тому обмеження на кількість учасників спілкування йшли в минуле. Одночасно, все більше уваги зверталось на графічну складову чат-додатку. 1990-ті роки були часом бурхливого розвитку онлайн-чатів. Саме тоді виникла легендарна програма для обміну миттєвими повідомленнями ICQ (фонетично співзвучна з англійською фразою I seek you – я тебе шукаю).

Створена у 1996 році ізраїльською компанією Mirabilis вона була централізованою службою миттєвого обміну повідомленнями, що використовувала протокол OSCAR. Користувач служби працював через додаток, який було запущено на пристрої підключеного до інтернету. Додаток підключався до серверу і через нього здійснювався пошук і зв'язок з іншими користувачами. Проте, обмін повідомленнями був можливий не лише через використання центрального серверу, а і без його участі. Як і в більшості високо навантажених мережевих систем сервер був не єдиним, а іноді сервер був лише кластером серверів. Таким чином, ICQ можна вважати першим в історії месенджером. Проте навіть в нього були недоліки, такі як обмеження на максимально допустиму довжину повідомлення. Але з розвитком доступності інтернету, його доступності та здешевлення і ці проблеми зникнуть.

З появою соціальних мереж, таких як MySpace та Facebook, спілкування в інтернеті знову вийшло на новий рівень. Доказом потреби людей в спілкуванні може бути те, що основною метою Марка Цукерберга при створенні Facebook був зв'язок між студентами американських коледжів.

Логічним продовженням розвитку засобів спілкування, з урахуванням швидкості мобільного інтернету та доступності смартфонів, було виникнення месенджерів, які встановлювались на телефонах. Демонстрація наскільки важливим для людей є спілкування з використанням соціальних мереж та месенджерів на телефонах зображена на рис. 1.1 – популярність категорій додатків встановлених на телефоні в США за вересень 2019.

У наш час існує доволі велика кількість месенджерів, які, незважаючи на відмінності, слугують одній меті – спілкуванню. Наразі, месенджери дуже широко розповсюджені, і підтвердження цьому можна побачити на рис.1.2, на якому продемонстрована кількість користувачів за жовтень 2019.

Рисунок 1.1 – Популярність категорій мобільних додатків

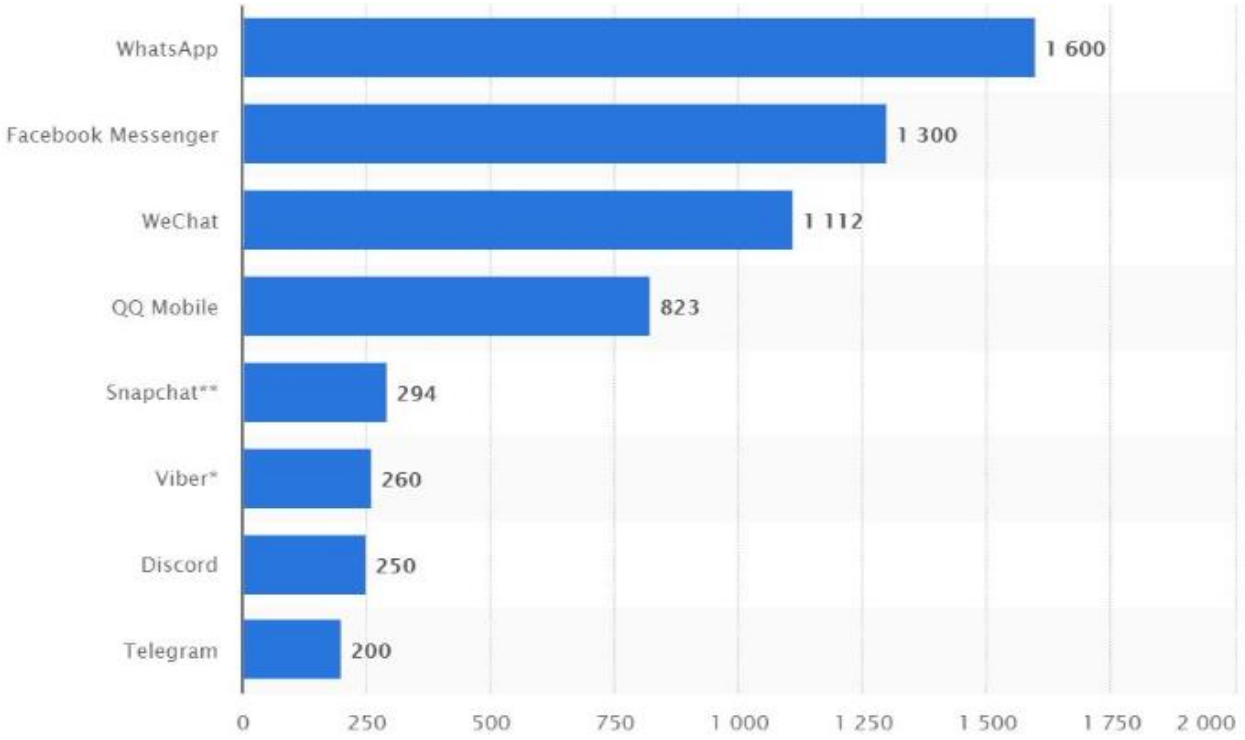


Рисунок 1.2 – Популярність месенджеров

Цікавою особливістю є кількість користувачів WeChat, який має широке розповсюдження переважно в Китаї. Для порівняння, яке покаже як глибоко ці засоби зв’язку увійшли в повсякденне життя: кількість користувачів месенджеру 1 мільярд 112 мільйонів, а кількість мешканців Китаю – 1 мільярд 387 мільйонів.

Телеграм є одним з найбільш швидко зростаючих месенджерів – кожного щорічно демонструє зріст на 50%. При цьому поступово він набуває популярності у нових країнах: наприклад, під час тимчасовій заборони WhatsApp в Бразилії в перші 24 години Телеграм було завантажено 5,7 мільйонів разів. Його ключовими особливостями є найбільша швидкість передачі повідомлень серед інших месенджерів, можливість копіювання та вставлення фотографій, відео, геолокацій, документів. При чому всі ці файли ефективно стискаються, а верхня границя величина файлу є 1.5 Гб. Також треба підкреслити можливість створення секретних чатів до змісту яких майже неможливо отримати доступ.

Отже, як показано, месенджери поступово входять в життя кожного, при чому використовують їх не лише для текстового спілкування, а й для аудіо та відео дзвінків. Тобто вони поступово займають нішу мобільних операторів. Оскільки Телеграм є одним з найбільш зростаючих месенджерів, то потреба розвитку чат-ботів є актуальною.

1.2 Застосування чат-ботів для бізнесу

Месенджери стають все дедалі більш розповсюдженими, а кількість користувачів росте з року в рік. Проте, якщо раніше було розглянуто лише їх застосування для спілкування між людьми, то доцільно розглянуто процес взаємодії бізнесу з користувачем за допомогою месенджерів.

Боти з’явились не лише в месенджерах, а і на звичайних сайтах, де вони були помічниками користувача. Відтепер достатньо було написати боту, і вже він переправить повідомлення відповідному агенту служби підтримки. Також бот міг бути електронними асистентом, за допомогою якого автоматизовувалась документація розпорядку, нагадування о зустрічах, тощо. Також не будемо забувати о можливостях голосових асистентів, таких як Siri, Alexa, Google Assistant чи Cortana від Apple, Amazon, Google та Microsoft відповідно.

Окремий різновид ботів, який використовуються в месенджерах – це чат-боти. Вони можуть мати різні цілі, такі як ігри, інформування чи відстежування чогось – наприклад акцій чи білетів на поїзд.

Тож, чат-боти насправді можуть покращувати і полегшувати щоденні справи завдяки автоматизації. Свого часу виникли CRM-системи які переслідували саме цю мету. А оскільки розповсюдження месенджерів стає дедалі більшим, то логічним продовженням розвитку чат-ботів може бути саме дублювання функцій CRM-систем, а саме частину взаємодії бізнесу та кінцевого користувача, адже користування приватним кабінетом в таких системах зручно лише при використанні на комп’ютері чи ноутбуці. Іншим шляхом може бути створення власного мобільного додатку для дублювання, але це вартує часу та коштів і виникають проблеми його підтримку для різних операційних систем та їх версій. Створення свого чат-боту відразу дає користувачу зручний та швидкий інтерфейс, а також проблема крос-платформової підтримки лягає на розробника месенджера. Також бізнес зі створенням свого бота одразу стає ближчим до користувача, адже додаток месенджера в нього вже встановлений і йому потрібно лише почати цим ботом користуватись.

Цікавим моментом може бути створення не власне бота для конкретної системи, а щось на кшталт конструктора, за допомогою якого можна створити чат-бота для власних потреб за допомогою лише налаштувань і уникнути процесу розробки. Також такий бот, окрім доволі зручного налаштування, має бути легким у підтримці та розвитку. Тобто при оновленні CRM-системи для чат-боту буде достатньо лише змінити налаштування.

Багато CRM-систем працюють використовуючи архітектурний стиль REST, а отже якщо навчити бота використовувати вже існуючи ресурсні ідентифікатори CRM-системи, та щоб він знав яку саме інформацію показувати користувачеві з отриманої відповіді, то проблема легкого створення бота для автоматизації взаємодії користувача стане ближчою до вирішення.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

2.1 Формування функціональних вимог

Функціональні вимоги описують поведінку системи, що відноситься до функціональності системи – це те, що система повинна робити. Враховуючи це, списком вимог до системи створення бота буде:

- система повинна приймати на вхід файл конфігурації у форматі yaml/yml;
- система повинна вміти отримувати з файлу дерево пунктів меню;
- система повинна вміти обробляти поле url з пункту меню з урахуванням ключових значень, які потрібно в ньому заповнити;
- система повинна вміти обробляти поле message з пункту меню з урахуванням ключових значень, які потрібно в ньому заповнити;
- система повинна вміти знаходити ключові поля з об’єктів, які надходять з серверної частини CRM-системи;
- система повинна бути інтегрованою з Telegram за допомогою Telegram Bot API;
- система повинна вміти отримувати повідомлення з CRM-системи та відправляти їх користувачу.

Список функціональних вимог до бота, інтегрованого з CRM-системою:

- система повинна авторизовувати користувача за номером телефону;
- система повинна давати змогу користувачу підключати та відключати тарифні плани;
- система повинна давати змогу користувачу підключати та відключати сервіси;
- система повинна давати змогу користувачу використовувати бонуси у програмі лояльності;

- система повинна давати змогу користувачу дивитись список підключених тарифів та сервісів;
- система повинна давати змогу користувачу дивитись історію отриманих бонусів;
- система повинна давати змогу користувачу отримати поточний стан рахунку та кількість бонусів;
- система повинна мати змогу отримувати ззовні повідомлення користувачу та відправляти їх через Telegram.

2.2 Формування нефункціональні вимог

Нефункціональні вимоги описують характеристику продуктивності системи, її якості, безпечності, масштабованості – це те, якою система має бути. Отже, списком нефункціональних вимог до системи створення бота буде:

- система повинна отримувати дані про клієнта з серверної частини CRM-системи та зберігати його у внутрішній базу даних;
- система повинна не зберігати застарілі дані та дані, що вже не використовуються у базі даних;
- система повинна валідувати те, що клієнт вводить та сповіщати у разі неправильно введених даних;
- система має підтримувати зв’язок з CRM-системою за допомогою REST архітектури;
- система повинна надавати простий та зрозумілий шаблон для налаштування.

3 ВИБІР ТА ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

Вибір технологій, на яких буде встановлено проект є дуже важливим питанням, адже від цього буде залежати вигляд системи та шляхи, якими компоненти системи будуть обмінюватися даними.

1) Для початку проаналізуємо, як саме створена система буде інтегрована з месенджером Telegram. Система Telegram пропонує зручний програмний інтерфейс, за допомогою якого створений бот буде мати можливість отримувати повідомлення та надсилати у відповідь – Telegram Bot API. Також бот зможе отримувати інформацію про користувача, з яким спілкується.
2) Тепер час обрати мову програмування. Оскільки для зв’язку з системою Telegram було обрано Telegram Bot API, то вибирати потрібно буде з мов, які він підтримує. Одна з них – це Java. Вона обрана через те, що це потужна високорівнева об’єктно-орієнтована мова програмування, яка використовується в більшості серверних частинах проектів. Важливою особливістю є те, що вона є багатоплатформовою, завдяки тому, що при компіляції коду вона не перетворює його одразу в машинний код, а в так званий байт-код. Далі, за допомогою інтерпретатора, цей код вже переводиться в машинний. Завдяки цьому написаний код буде виконуватися однаково у всіх системах.
3) Для обраної мови Java існує безліч бібліотек та фреймворків. Spring Framework один з них. Це програмний фреймворк з відкритим кодом для платформи Java. Він містить багато модулів, які дають готові рішення відомих проблем та суттєво спрощують розробку. З його використанням зникає необхідність написання дублюючого коду, та можна зосередитись на реалізації бізнес-логіки додатку. Фреймворк реалізує шаблон розробки Dependency Injection та принцип розробки Inversion of Control. Завдяки імплементації цих шаблонів система будується за принципами SOLID та має низьку зв’язність, що полегшує подальше

розширення функціоналу. В цьому проекті були використані наступні модулі фреймворку:

- Spring Core – основний модуль фреймворку, який містить основні компоненти для побудови проекту – за допомогою них проект має структурно-цілісний вид;
- Spring MVC – модуль, який реалізує шаблон розробки Model View Controller, який використовується для створення ресурсних методів, за допомогою яких компоненти проекту обмінюються інформацією;
- Spring Data – модуль, який полегшує використання баз даних, оскільки бере на себе всю відповідальність за відкриття/закриття сесій, а для найбільш типових запитів до БД існують готові рішення, а отже суттєво прискорюється процес розробки.

4) Для спрощення розгортання та запуску додатку було обрано Spring Boot – це бібліотека інтегрована зі фреймворком Spring. Основним плюсами використання є те, що вона сама управляє версіями необхідних бібліотек, а отже не виникає проблеми, коли версії не підходять одна до одної. Також включає в себе сервер додатків Tomcat.
5) Оскільки основною метою CRM-системи є зберігання інформації, то потрібно визначитися як саме система буду інтегрована з базою даних. По замовчуванню Java зв'язана з базами даних за допомогою JDBC – стандартом взаємодії системи з базою даних, який заснований на використанні драйверів для зв'язку з базами від різних провайдерів. Таким чином написаний код працює однаково на різних базах даних. Проте в цьому підході є вагомих недолік – велика кількість дублюючого коду, такого як відкриття та закриття сесії, в рамках яких виконуються запити до бази або налаштування для під’єднання до бази. Для оптимізації взаємодії системи та бази даних було обрано Hibernate, який сам візьме відповідальність на виконання однотипних операцій. Це фреймворк, за допомогою якого реалізовано зіставлення таблиці в БД зі

звичайними Java об’єктами. Через це спрощуються процес розробки, оскільки маніпулювати CRUD операціями стає легше.

6) Тепер слід визначитися з провайдером бази даних. На початку розробки зручно використовувати таку базу, яка зберігає інформацію лише під час роботи системи, а після закінчення роботи всі дані видаляються – це зручно для тестування, а також не потребує великої кількості налаштувань. Один з таких провайдерів баз – це H2. Він обраний не тільки через те, що має режим збереження даних в пам’яті програми при її виконанні, але і те, що також може бути налаштований на зберігання даних і після завершення роботи системи. Через це він є зручним в розробці концепту майбутньої програми.
7) Оскільки при розробці системи ми будемо користуватися сторонніми бібліотеками, то потрібно вибрати як буде керуватись додання цих бібліотек. Одним з сучасних методів, за допомогою яких бібліотеки будуть завантажуватися це Apache Maven – засіб автоматизації роботи з програмними проектами, який використовується для завантаження сторонніх бібліотек та для збірки проекту. Також, завдяки налаштуванням, може під час збірки проекту виконувати дії спрямовані на автоматизацію процесу розробки: запускання тестів чи генерація Java класів з текстових файлів або запуск скриптів.
8) Оскільки майбутня система має бути якісною створеною та без помилок, то для запобігання виникнення помилок потрібно реалізувати тести. Так як було обрано Java, як мову програмування, то за засіб тестування буде обрано JUnit – фреймворк, який використовується для модульного тестування додатку. Також для полегшення тестування було обрано бібліотеку Mockito.
9) Навіть в якісно створеній системі існує вірогідність виникнення помилок, то для їх знаходження розробникам можуть допомогти логи. Для цього існують спеціальні файли куди записуються повідомлення про дії, які система виконує. Для спрощення логування було обрано

бібліотеку Log4j. Важливою особливість є те, що за допомогою налаштувань можна задати формат логів, їх рівень (за допомогою цього можна фільтрувати повідомлення по їх важливості).

10) Оскільки створений бот повинен мати можливість бути налаштованим за допомогою зовнішніх налаштувань, то для зберігання цих налаштувань зручно обрати формат файлу yaml. Для взаємодії з файлами цього формату оберемо бібліотеку Snake Yaml, за допомогою якої зміст файлу буде зіставлятися з звичними Java об'єктами.
11) Наостанок потрібно встановити де саме система буде розроблена. На сьогодні найбільш розвиненими є Eclipse та IntelliJ IDEA. І хоча повна версія Eclipse є безкоштовною, проте IDEA має більш зручний інтерфейс. Також вона має безкоштовну версію з трохи усіченим функціоналом.

4 СТРУКТУРНА СХЕМА СИСТЕМИ

Після визначення технологій, на яких буде створена система, можна спланувати її структуру. Для початку розглянемо Telegram Bot API.

Bot API – це те, за допомогою чого створений бот буде під'єднаний до системи Telegram. У цій системи бот – це спеціальний обліковий запис, через який чат-бот, що буде створений зможе приймати повідомлення та відповідати на них.

Сама система використовує внутрішній протокол шифрування MTProto, проте спілкування бота з системою буде через проміжний сервер, який обробить всі шифрування. Для зв'язку з цим сервером використовується HTTPS-протокол.

Для отримання оновлень бот може використовувати веб-хуки – обробники подій. Бот підписується на оновлення та при наявності необроблених повідомлень одразу їх отримує завдяки веб-хукам. Повідомлення зберігаються на сервері на протязі 24 годин і за цей час бот повинен їх прочитати.

Оскільки створений чат бот буде інтегрованим не тільки з системою Telegram, а і з серверною частиною CRM-системи, то структура всієї системи буде складатись з чотирьох компонентів: клієнтської машини, серверу Telegram Bot, чат-боту та серверної частини CRM-системи. Функціональну схему цієї системи наведено у Додатку А.

Як видно, повідомлення від користувача буде потрапляти у сервер Telegram Bot, звідки, завдяки веб-хукам, його буде отримувати чат-бот, обробляти та надсилати запит у серверну частину, якщо це буде потрібно. Сама серверна частина буде мати власну базу даних та буде побудована згідно шаблону MVC. Так як заплановано що бот теж повинен вміти приймати запити з серверної частини, то його теж буде побудовано за шаблоном MVC.

Оскільки було обрано цей шаблон, то зручно буде побудувати систему, використовуючи архітектурний стиль REST. Отже, серверна частина повинна

мати ресурсні ідентифікатори до яких буде «звертатися» чат-бот. Так само і бот повинен мати хоча б один ресурсний ідентифікатор, що приймати повідомлення з CRM-системи. REST працює поверх транспортного протоколу HTTP, отже для спілкування між собою компоненти будуть використовувати саме його.

Чат-бот також буде мати власну базу даних, яку буде використовувати для збереження користувачів після авторизації – таким чином, користувачу потрібно буде лише один раз авторизуватися в системі. Запити у власну базу обидві системи будуть робити користуючись фреймворком Hibernate, який є реалізацію ORM – технологією програмування, яка зв'язує базу даних з об'єктно-орієнтованим сутностями.

Оскільки було обрано Spring Boot, то обидва компоненти системи будуть використовувати сервери додатків Tomcat, який постачається з ним.

5 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

Загальна картина існуючих сценаріїв зображено в Додатку Л. Нижче більш деталізовані сценарії.

5.1 UML-схема перегляду та активація тарифного плану

На рис. 5.1 зображено сценарій за яким користувач може переглянути список доступних тарифних планів та активувати обраний (якщо він ще не активний). Користувач має бути підписаним на використання бота і бути в ньому авторизованим. Авторизація відбувається шляхом отримання користувача з серверної частини CRM-системи. Якщо користувача знайдено, то бот отримує його з зовнішньої системи та зберігає у власну базу даних. Після цього авторизація буде більше не потрібна. Якщо користувача не знайдено, то йому покажеться відповідне повідомлення з посиланням на реєстрацію в системі. Отже, авторизований користувач може переглянути всі категорії тарифних планів та перейти на потрібну. Після цього йому покажеться список всіх тарифів з цієї категорій. Він може активувати тарифний план шляхом вибору потрібного. Система чат-боту надішле запит на серверну частину CRM-системи. При цьому на серверній частині відбуваються такі дії:

- перевірка, що тарифний план ще не підключено;
- перевірка балансу користувача – порівняння залишку на рахунку та вартості тарифного плану;
- додавання сутності тарифу до сутності користувача зі збереженням в базі даних;
- оновлення стану рахунку користувача зі збереженням в базі;
- надсилання статус про успішність операції.

В залежності від статусу повідомлення користувачеві показується відповідне повідомлення.

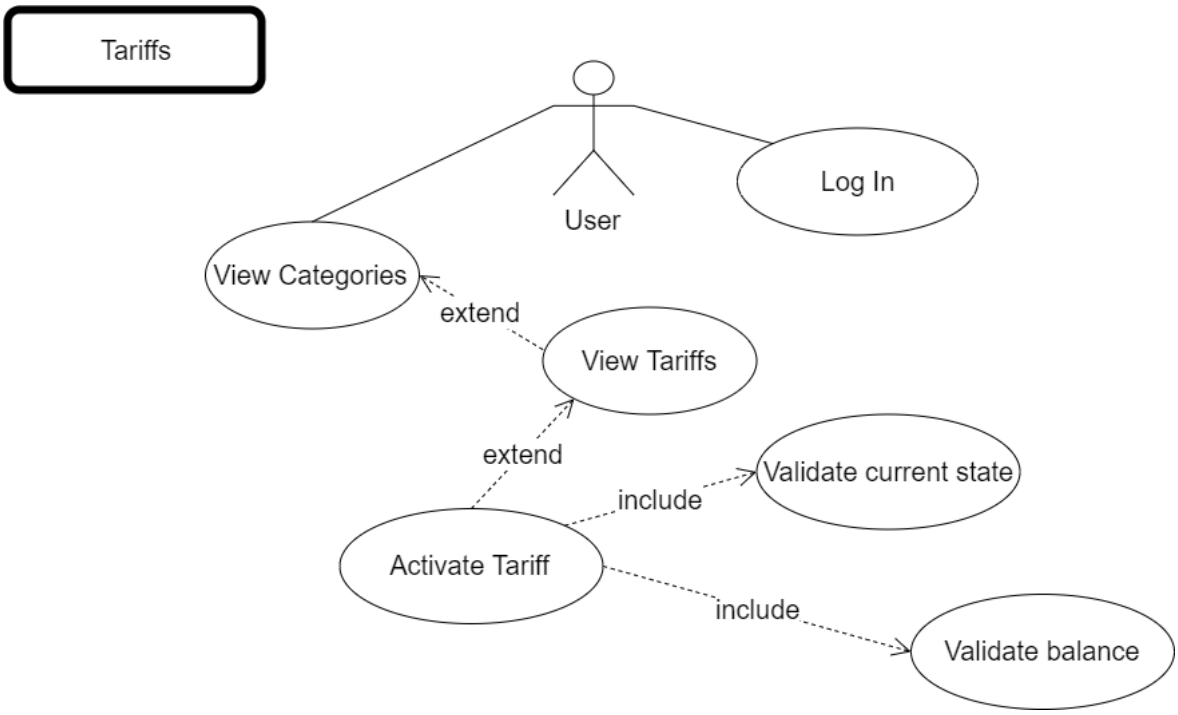


Рисунок 5.1 – Схема перегляду та активації тарифного плану

5.2 UML – схема перегляду та деактивації тарифного плану

На Рис 5.2 зображено сценарій за яким користувач може переглянути список доступних тарифних планів та деактивувати обраний (якщо він активний). Користувач має бути підписаним на використання бота і бути в ньому авторизованим. Авторизація відбувається шляхом отримання користувача з серверної частини CRM-системи. Якщо користувача знайдено, то бот отримує його з зовнішньої системи та зберігає у власну базу даних. Після цього авторизація буде більше не потрібна. Якщо користувача не знайдено, то йому покажеться відповідне повідомлення з посиланням на реєстрацію в системі. Отже, авторизований користувач може переглянути всі категорії тарифних планів та перейти на потрібну. Після цього йому покажеться список всіх тарифних планів з цієї категорій. Він може деактивувати тариф шляхом вибору потрібного. Система чат-боту надішле запит на серверну частину CRM-системи. При цьому на серверній частині відбуваються такі дії:

- перевірка, що тарифний план є активним

- видалення сутності тарифного плану з сутності користувача зі збереженням в базі даних;
- надсилання статус про успішність операції.

В залежності від статусу повідомлення користувачеві показується відповідне повідомлення.

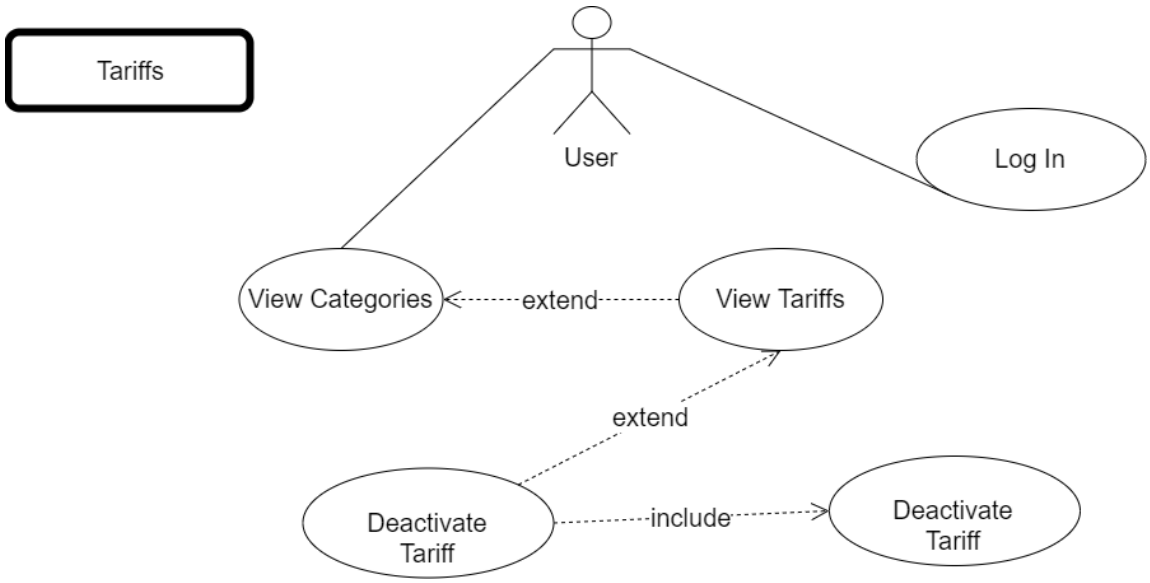


Рисунок 5.2 – Перегляд і деактивація тарифного плану

5.3 UML – схема перегляду та активації сервісу

На рис. 5.3 зображено сценарій за яким користувач може переглянути список доступних сервісів та активувати обраний (якщо він ще не активний). Користувач має бути підписаним на використання бота і буду в ньому авторизованим. Авторизація відбувається шляхом отримання користувача з серверної частини CRM-системи. Якщо користувача знайдено, то бот отримує його з зовнішньої системи та зберігає у власну базу даних. Після цього авторизація буде більше не потрібна. Якщо користувача не знайдено, то йому покажеться відповідне повідомлення з посиланням на реєстрацію в системі. Отже, авторизований користувач може переглянути всі доступні сервіси перейшовши на відповідний пункт меню. Він може активувати сервіс шляхом

вибору потрібного. Система чат-боту надішле запит на серверну частину CRM-системи. При цьому на серверній частині відбуваються такі дії:

- перевірка, що сервіс ще не підключено;
- перевірка балансу користувача – порівняння залишку на рахунку та вартості сервісу;
- додавання сутності сервісу до сутності користувача зі збереженням в базі даних;
- оновлення стану рахунку користувача зі збереженням в базі;
- надсилання статус про успішність операції.

В залежності від статусу повідомлення користувачеві показується відповідне повідомлення.

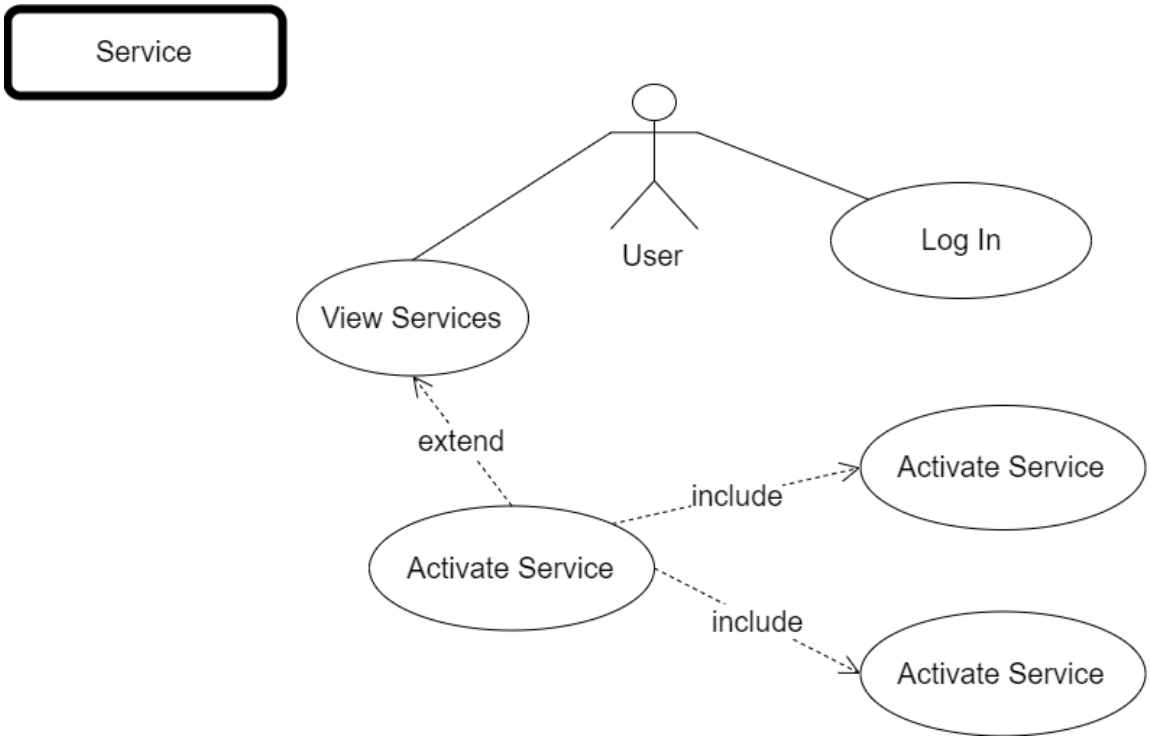


Рисунок 5.3 – Перегляд і активація сервісу

5.4 UML – схема перегляду та деактивації сервісу

На рис. 5.4 зображено сценарій за яким користувач може переглянути список доступних сервісів та деактивувати обраний (якщо він активний).

Користувач має бути підписаним на використання бота і буду в ньому авторизованим. Авторизація відбувається шляхом отримання користувача з серверної частини CRM-системи. Якщо користувача знайдено, то бот отримує його з зовнішньої системи та зберігає у власну базу даних. Після цього авторизація буде більше не потрібна. Якщо користувача не знайдено, то йому покажеться відповідне повідомлення з посиланням на реєстрацію в системі. Отже, авторизований користувач може переглянути всі доступні сервіси перейшовши на відповідний пункт меню. Він може деактивувати сервіс шляхом вибору потрібного. Система чат-боту надішле запит на серверну частину CRM-системи. При цьому на серверній частині відбуваються такі дії:

- перевірка, що сервіс є активним;
- видалення сутності сервісу з сутності користувача зі збереженням в базі даних;
- надсилання статус про успішність операції.

В залежності від статусу повідомлення користувачу показується відповідне повідомлення.

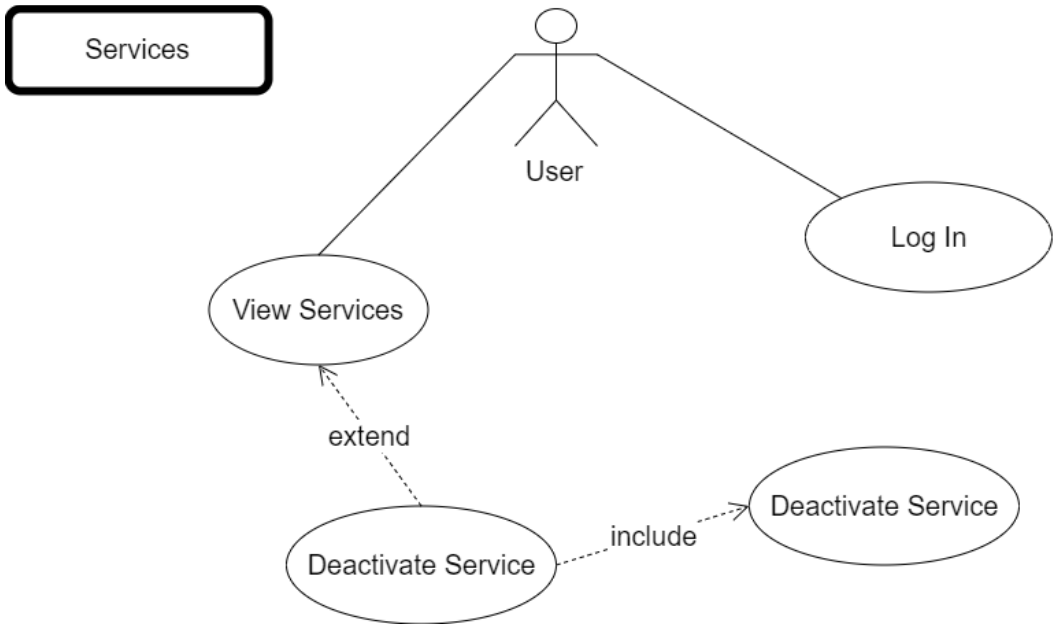


Рисунок 5.4 – Перегляд та деактивація сервісу

5.5 UML – схема перегляду та активації пропозицій програми лояльності

На рис. 5.5 зображено сценарій за яким користувач може переглянути список доступних пропозицій програми лояльності та активувати обрану. Користувач має бути підписаним на використання бота і буду в ньому авторизованим. Авторизація відбувається шляхом отримання користувача з серверної частини CRM-системи. Якщо користувача знайдено, то бот отримує його з зовнішньої системи та зберігає у власну базу даних. Після цього авторизація буде більше не потрібна. Якщо користувача не знайдено, то йому покажеться відповідне повідомлення з посиланням на реєстрацію в системі. Отже, авторизований користувач може переглянути всі доступні пропозиції програми лояльності перейшовши на відповідний пункт меню. Він може активувати пропозицію шляхом вибору потрібної. Система чат-боту надішле запит на серверну частину CRM-системи. При цьому на серверній частині відбуваються такі дії:

- перевірка бонусного балансу користувача – порівняння кількості бонусів та бонусну вартість пропозиції;
- додавання сутності пропозиції до сутності користувача зі збереженням в базі даних;
- оновлення стану бонусного рахунку користувача зі збереженням в базі;
- надсилання статус про успішність операції.

В залежності від статусу повідомлення користувачеві показується відповідне повідомлення.

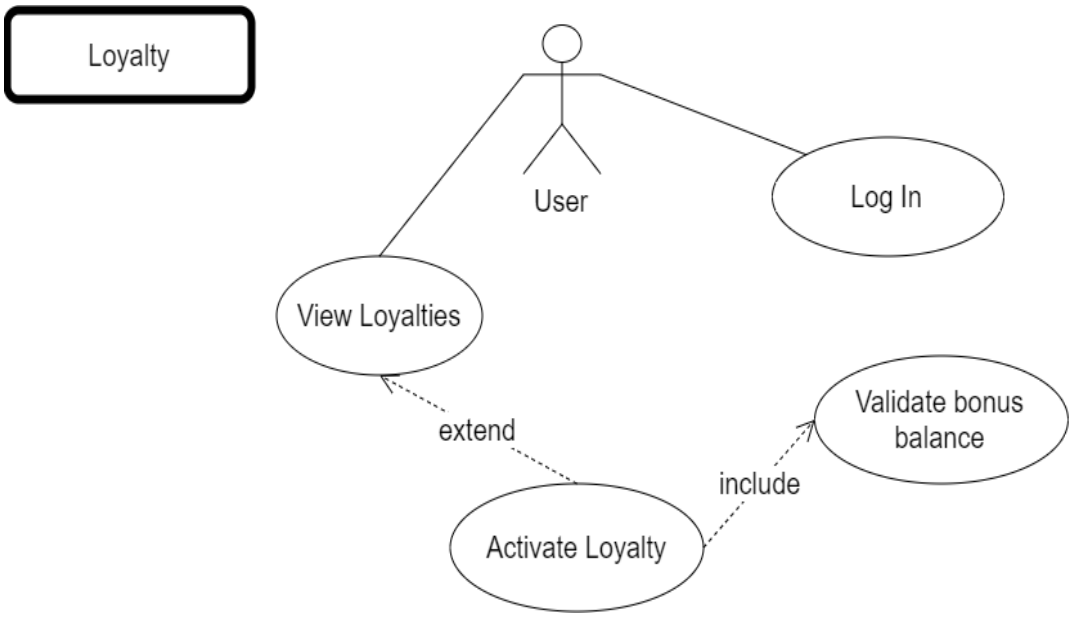


Рисунок 5.5 – Перегляд та активація пропозиція програми лояльності

5.6 UML – схема перегляду історії бонусів, підключених тарифів, сервісів та стану рахунку

На рис. 5.6 зображено сценарій за яким користувач може переглянути список доступних пропозицій програми лояльності та активувати обрану. Користувач має бути підписаним на використання бота і буду в ньому авторизованим. Авторизація відбувається шляхом отримання користувача з серверної частини CRM-системи. Якщо користувача знайдено, то бот отримує його з зовнішньої системи та зберігає у власну базу даних. Після цього авторизація буде більше не потрібна. Якщо користувача не знайдено, то йому покажеться відповідне повідомлення з посиланням на реєстрацію в системі. Отже, авторизований користувач може переглянути всю інформацію щодо підключених тарифів та сервісів, активованих пропозицій лояльності та стану рахунку та бонусного рахунку перейшовши у пункт меню з налаштуваннями. Система чат-боту надішле запит на серверну частину CRM-системи. Вона отримає потрібні сутності з бази даних та надішле їх у відповідь. Після цього користувачу прийде повідомлення з запитуваною інформацією.

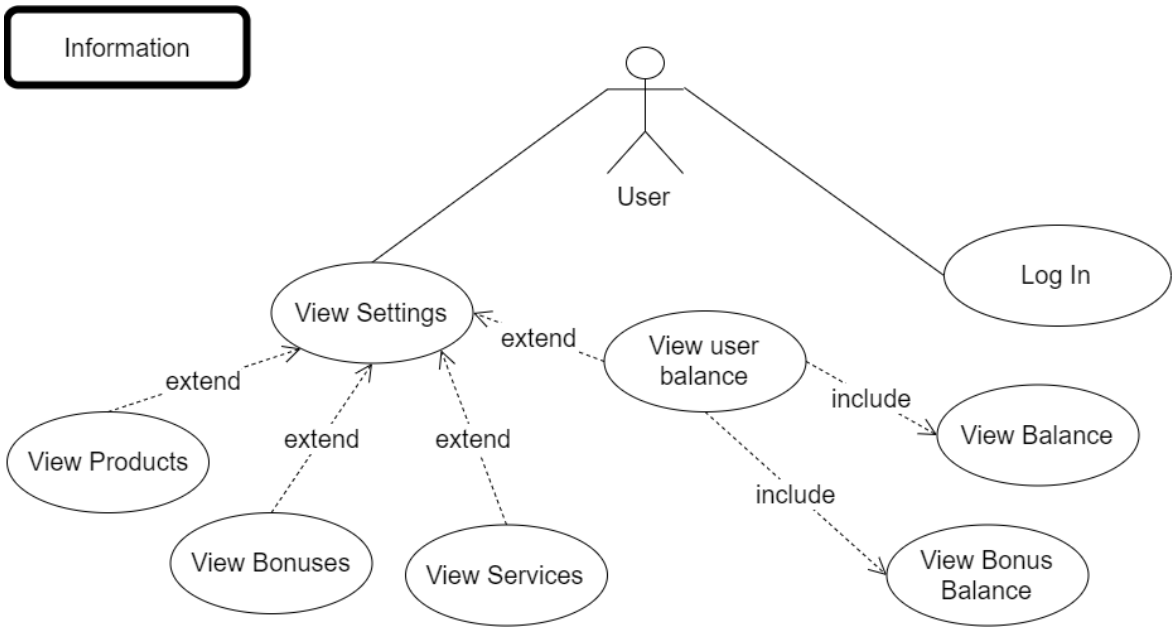


Рисунок 5.6 – Перегляд інформація користувача

6 РЕАЛІЗАЦІЯ ЕЛЕМЕНТІВ СИСТЕМИ

Оскільки для реалізації всієї системи повинно бути реалізовано дві компоненти: серверна частина CRM-системи і компонент чат бота. Загальний вигляд того, як ці дві компоненти діють та інтегруються наведено у Додатку Б. Нижче буде розглянуто реалізація класів цих компонент.

6.1 Реалізація елементів CRM-системи

Загальну структурну схеми серверної частини CRM-системи представлено у Додатку Г та Додатку Д.

Якщо розглянути більш поверхнево, то система побудована за шаблоном MVC. Кореневими елементами системи є:

- контролери;
- репозиторії;
- моделі;
- менеджери;
- моделі представлення.

6.1.1

Контролери – потрібні для отримання запитів ззовні та надсиланні відповідей. Оскільки в система побудовано згідно стилю архітектури REST, то методи контролера повинні мати унікальні ідентифікатори ресурсу, які задаються унікальною url. Також кожен ресурсний метод повинен бути налаштований на приймання конкретного HTTP методу. Зазвичай це методи: GET, POST, PUT, DELETE. Контролери даної системи зображені на рис.6.1.

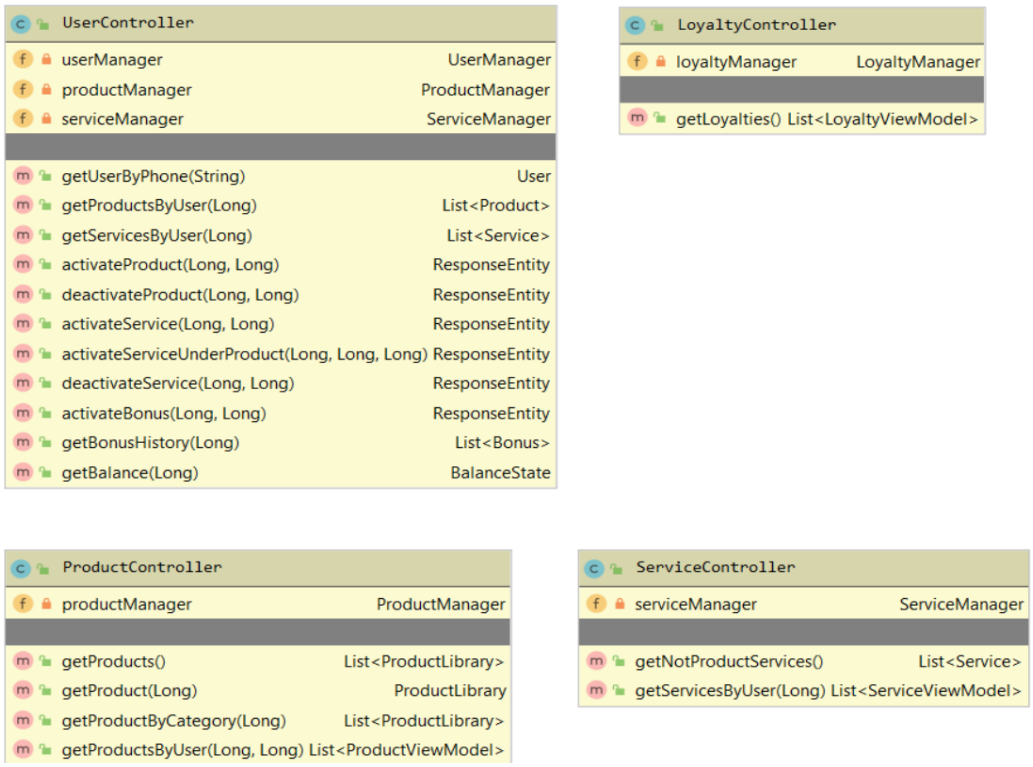


Рисунок 6.1 – Контролери

6.1.2 Менеджери – це рівень сервісів у системи. Вони потрібні для отримання даних з рівню репозиторіїв та виконання операцій над ними. Оскільки в методі менеджеру можуть виконані декілька операцій над базою даних, то саме на цьому рівні зручно імплементувати транзакційність запитів. Також класи цього рівня конвертують сутності з бази до моделей представлення. Менеджери даної системи зображені на рис.6.2.

6.1.3 Репозиторії – це рівень, який працює з базою даних. Оскільки використовувався фреймворк Hibernate, то деякі репозиторії виглядають порожніми. Насправді, вони імплементують стандартну поведінку для JPA репозиторіїв та можуть виконувати прості CRUD операції. Так як використовувався Spring Data, то репозиторії можуть формувати запити до бази даних лише завдяки назві методу. Для специфічних запитів було використано анотацію Query. Репозиторії даної системи зображені на рис.6.3.

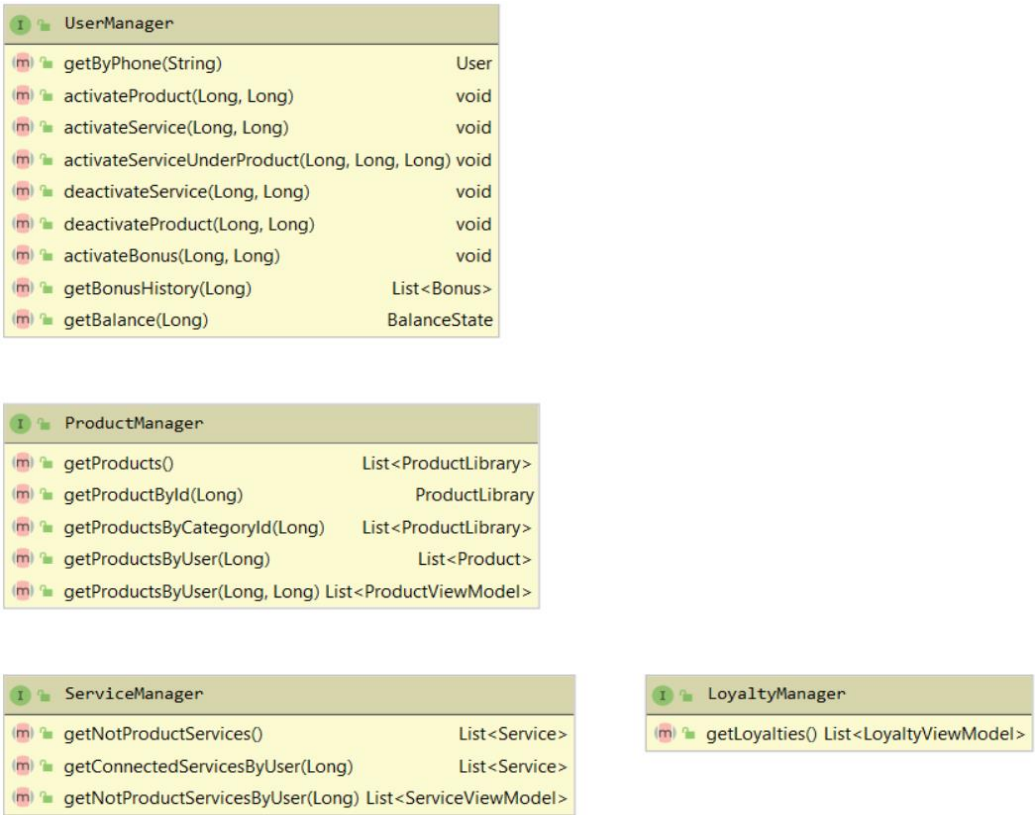


Рисунок 6.2 – Менеджери

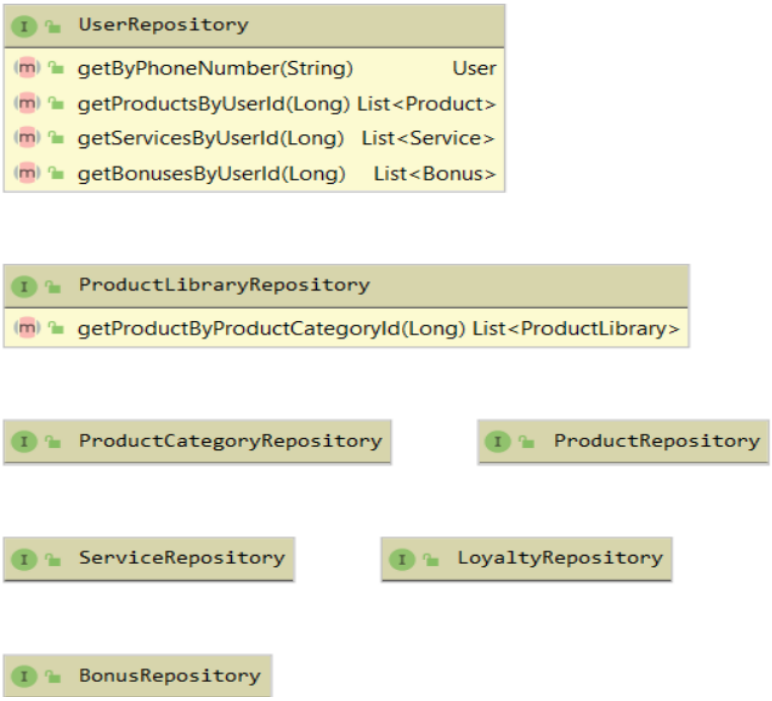


Рисунок 6.3 – Репозиторії

6.1.4 Моделі – це Java-об’єкти, які є зображенням сутностей бази даних. Вони мають спеціальні анотації для класу та полів завдяки

яким проводиться зіставлення сутності та цього об’єкту. Також містять зв’язки між таблицями, такі як один до одно, один до багатьох, багато до багатьох. Моделі системи зображені на рис.6.4.

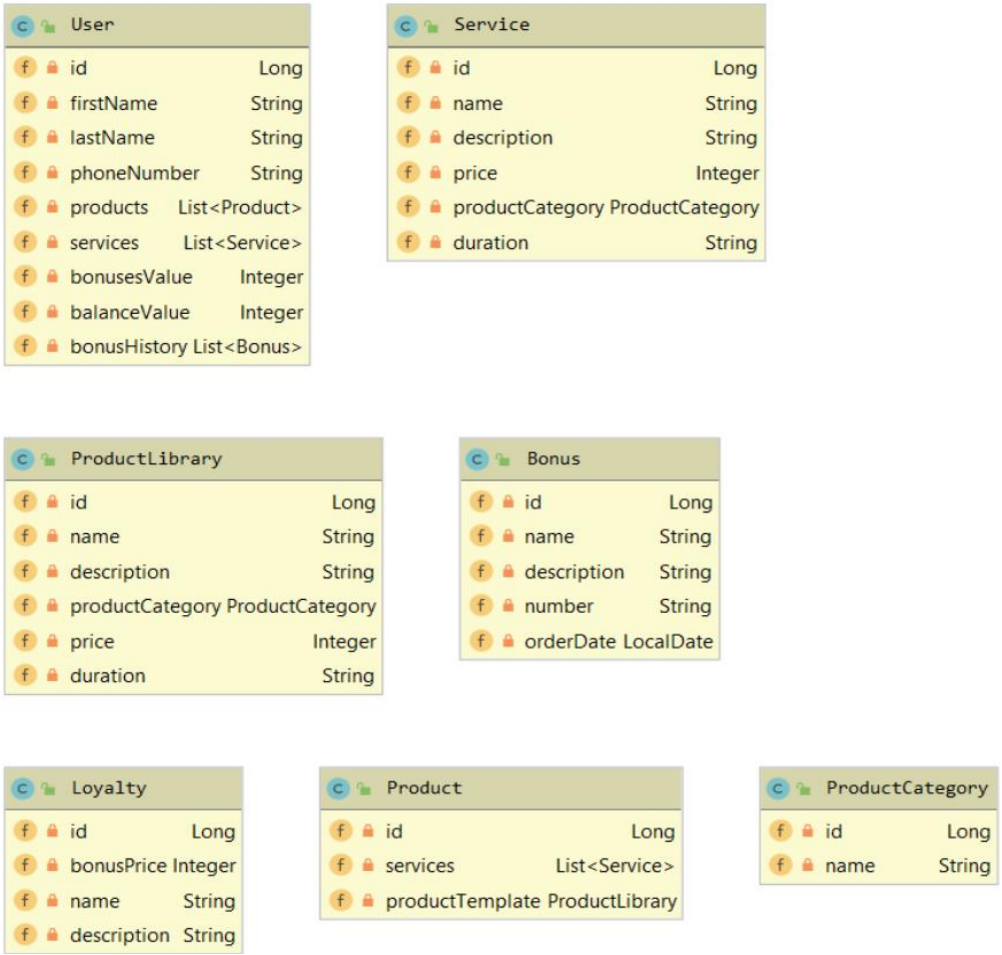


Рисунок 6.4 – Моделі

6.1.5 Моделі представлення – саме них зазвичай відправляють контролери у відповідь. Необхідні тому, що сутності з бази даних можуть мати надлишкову інформацію, яка не використовується в інших частинах системи. Тому менеджери конвертують звичайні моделі в моделі представлення де прибирають надлишкові дані та компонують деякі поля з інших моделей. Моделі представлення зображені на рис.6.5.

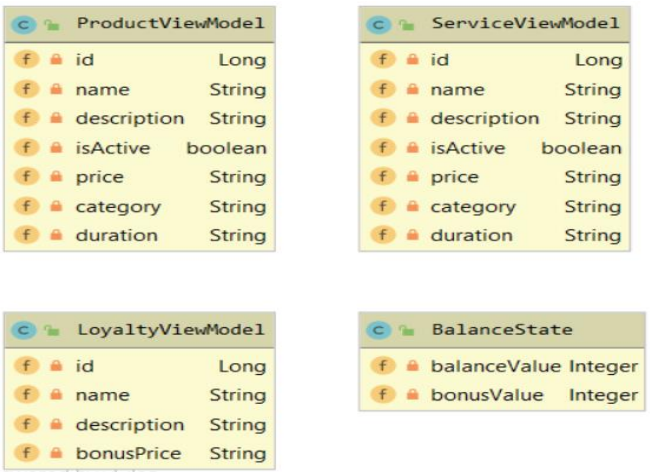


Рисунок 6.5 – Моделі представлення

6.2 Реалізація елементів системи чат-боту

Загальну структурну схеми чат-бота представлено у Додатку Ж. Якщо розглянути більш детально, то система чат-бота має такі основні компоненти:

- рівень бази даних;
- конфігурації;
- рівень REST;
- сервіси;
- класи-помічники;
- моделі;
- імплементація бота.

6.2.1 Рівень бази даних включає в себе об’єкти-сутності та репозиторії, необхідні для роботи з внутрішньою базою даних бота. Детальніше розглядались вище на прикладі CRM. Класи цього рівня зображені на рис.6.6.

6.2.2 Конфігурації – об’єкти, які створюються при старті додатку та заповнюються даними з файлу налаштувань. Містять всю необхідну інформацію для того, щоб бот був успішно налаштований. Конфігурації зображені на рис.6.7.

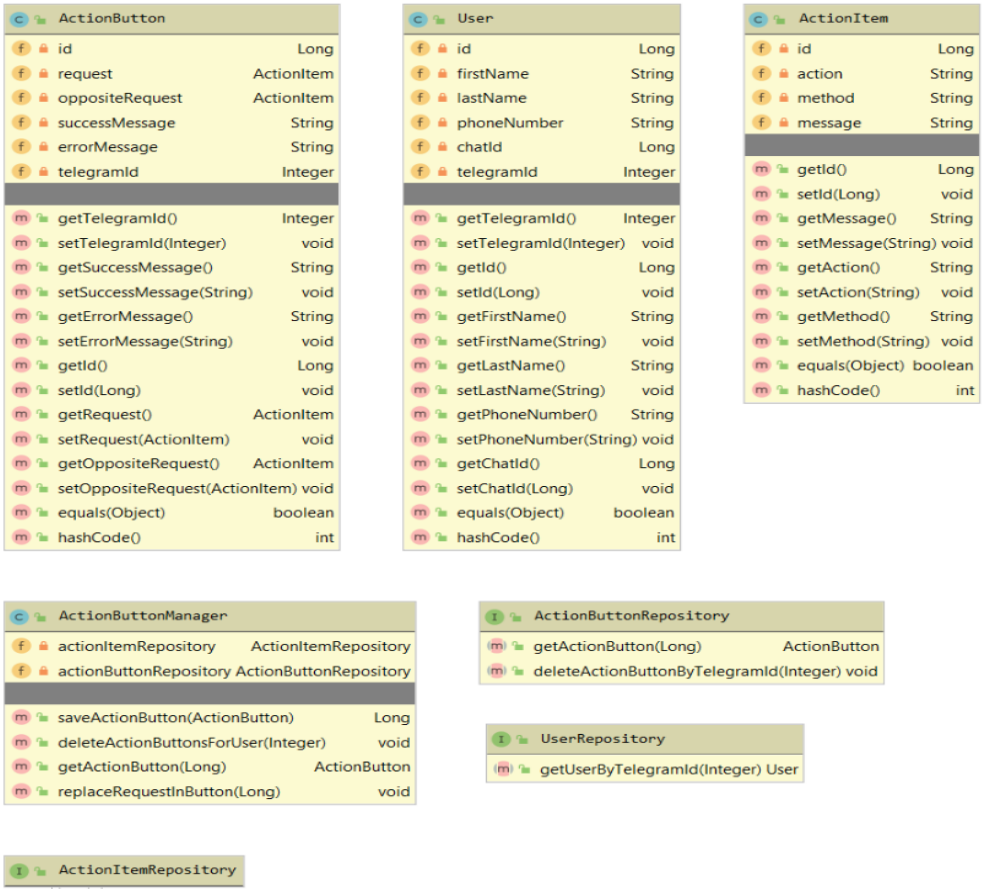


Рисунок 6.6 – Рівень бази даних

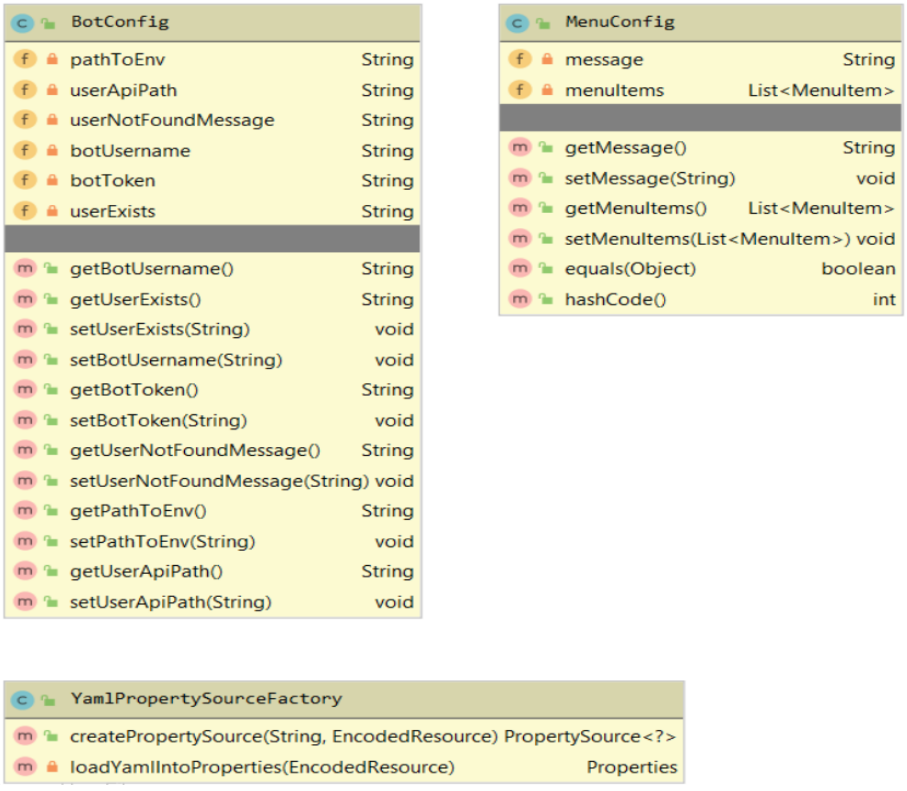


Рисунок 6.7 – Конфігурації

6.2.3 В рівень REST входять як і контролер, який приймає повідомлення ззовні, так класи, необхідні для відправлення повідомлень самим ботом. Зображені на рис.6.8.

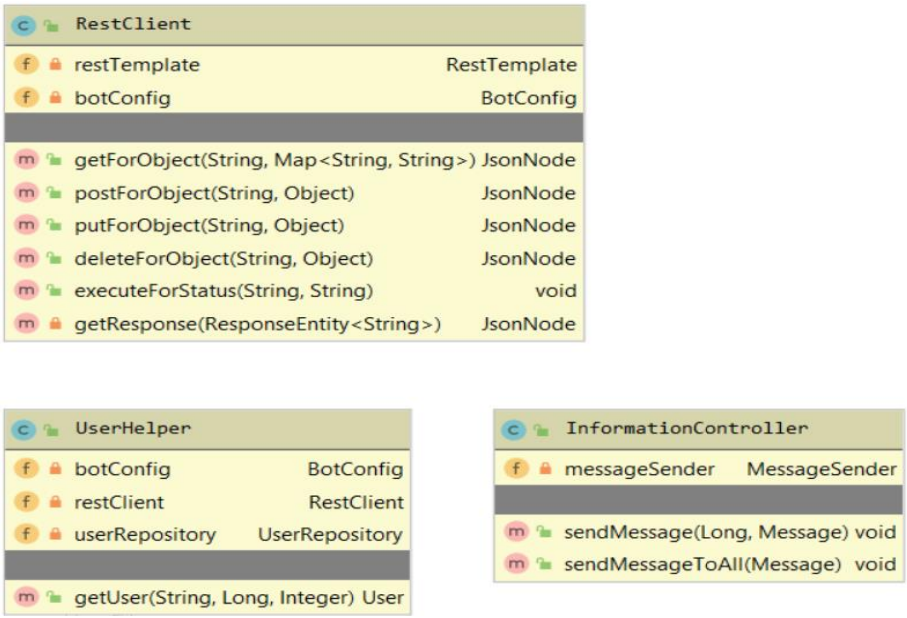


Рисунок 6.8 – Рівень REST

6.2.4 Сервіси в системі займаються створенням відповідей, які будуть надіслані ботом, роботою з пунктами меню та відправленням повідомлень до користувача, які прийшли ззовні. Зображені на рис.6.9.

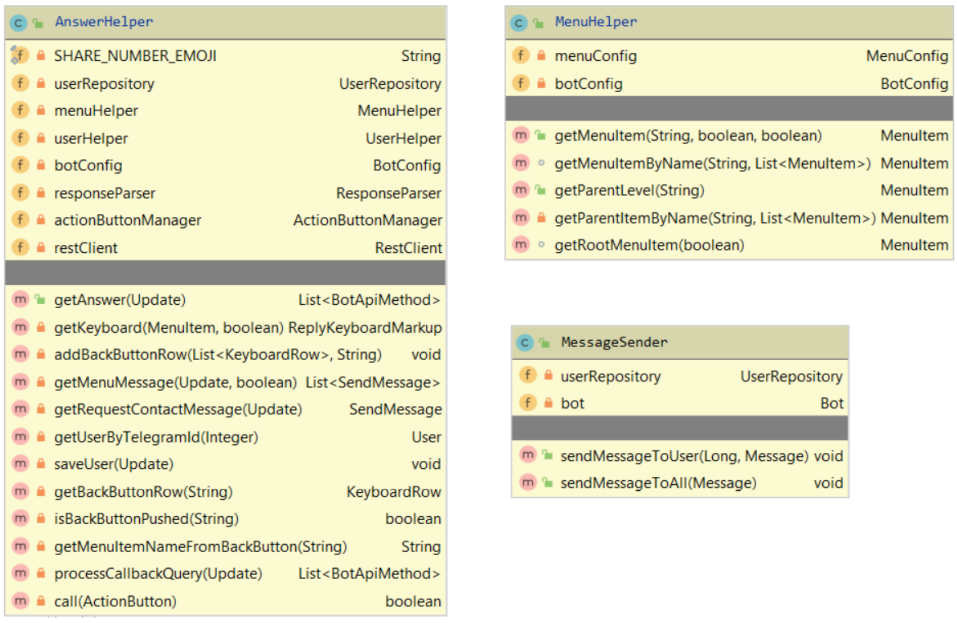


Рисунок 6.9 – Сервіси

6.2.5 В системі є два типу класів-помічників. перший – це абстрактні класи які містять утилітні методи, другий – для роботи з обробкою даних, які бот отримує з серверної частини CRM-системи. Зображені на рис.6.10 – рис.6.11.

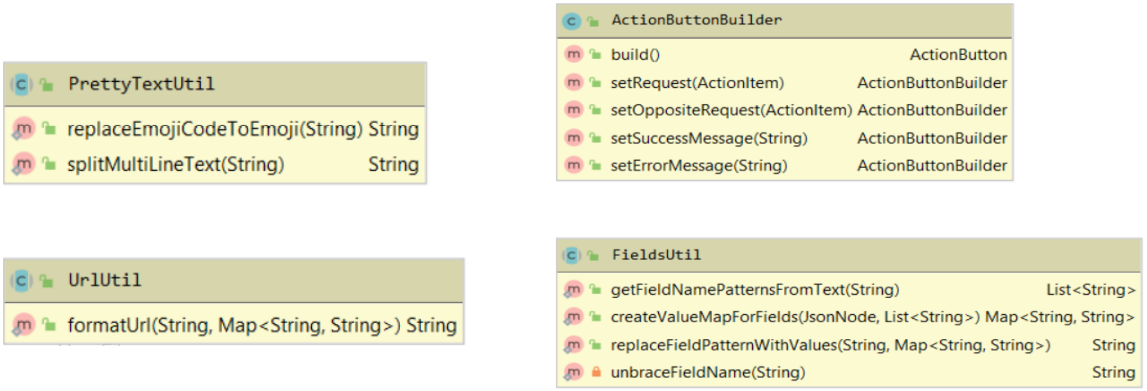


Рисунок 6.10 – Перший тип класів-помічників

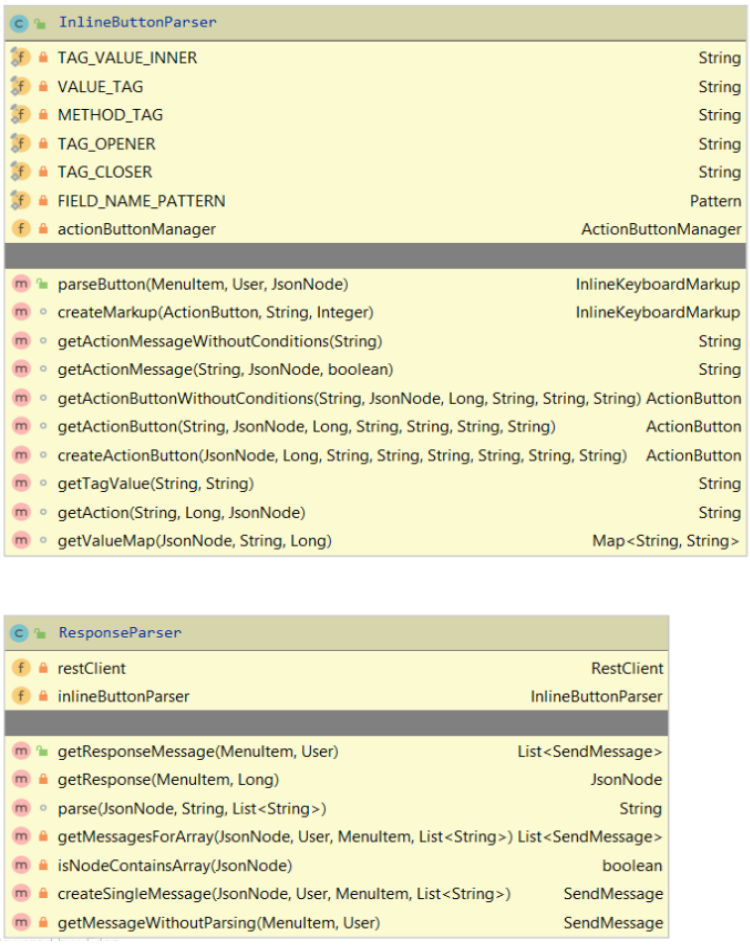


Рисунок 6.11 – Другий тип класів-помічників

6.2.6 Імплементація бота – це кореневий елемент системи. Цей адаптер потрібний для того, щоб бот міг відправляти повідомлення до серверу месенджеру. Для цього він унаслідується від TelegramLongPollingBot класу від Telegram Bot API та перевизначає методи, спрямовані на отримання та формування повідомлень, а також реєстрація имені та токену бота в системі Telegram. Зображено на рис.6.12.

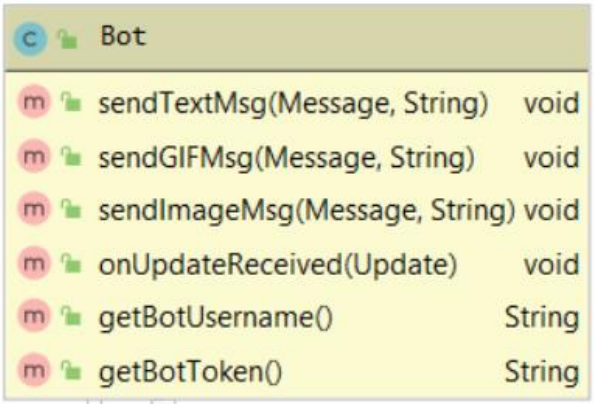


Рисунок 6.12 – Імплементація бота

7 ER-МОДЕЛЬ

ER-модель – це засіб опису моделі даних в системі. Вона зображає сукупність таблиці у базі даних та зв’язки між ними. Також кожна таблиця містить множину полів та їх тип і інформацію щодо ключів таблиці та індексів.

В нас є дві системи: демонстрація серверної частини CRM-системи та система для створювання ботів. Спочатку розглянемо структуру бази даних серверної частини CRM-системи. Вона складається з 11 таблиць. Повністю зображена у Додатку В.

На рис 7.1 зображено таблицю USER, що зберігає в собі інформацію про користувача:

- first_name – им’я користувача;
- last_name – прізвище користувача;
- phone_number – номер мобільного телефону користувача, який у подальшому з’єднує користувача між двома системами;
- balance_value – стан рахунку користувача;
- bonuses_value – стан бонусного рахунку користувача;
- id – первинний та зовнішній ключ таблиці.

Також таблиця зберігає історію отриманих бонусів, які задані відношенням багато до багатьох. Для цього слугує таблиця USER_BONUS_HISTORY, яка зв’язує користувачів та бонуси за допомогою двох зовнішніх ключів – унікальних ідентифікаторів таблиць USER та BONUS. Також таблиця USER містить зв’язки з підключеними продуктами та сервісами, які будуть розглянуті нижче.

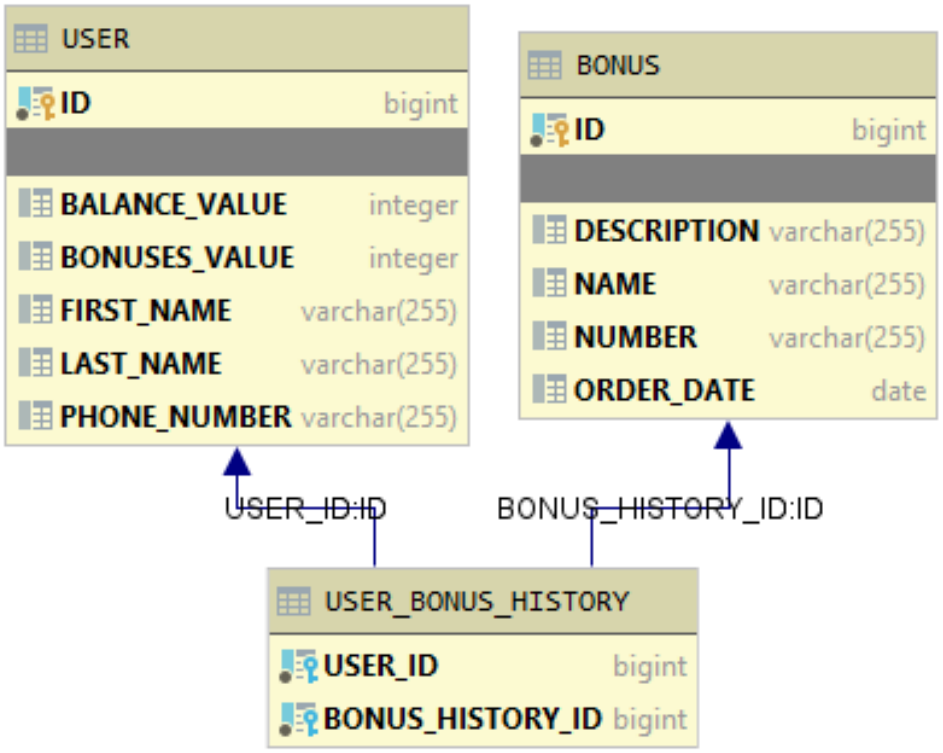


Рисунок 7.1 – Сутності User та Bonus

На рис. 7.1 зображено таблицю BONUS, що зберігає інформацію про підключену пропозицію програми лояльності:

- id – первинний та зовнішній ключ таблиці;
- description – опис пропозиції – що саме користувач підключає;
- name – им’я пропозиції програми лояльності;
- number – унікальний номер пропозиції;
- order_date – дата, коли пропозицію було підключено.

Таблиця PRODUCT_CATEGORY є таблицею-словником і містить опис різних категорії продуктів та сервісів (Phone plans, TV, Internet, OTT):

- id – первинний та зовнішній ключ;
- name – им’я категорії.

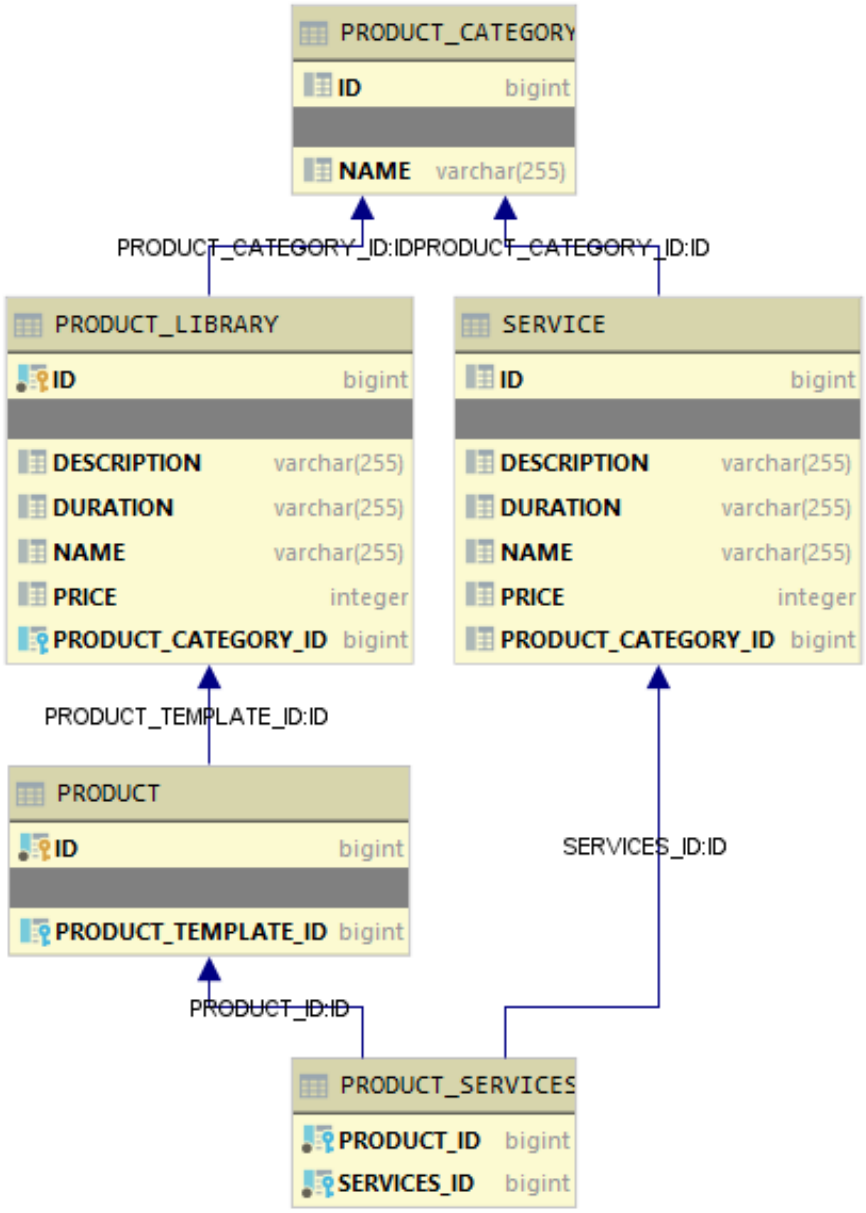


Рисунок 7.2 – Сутності Product, Service, Product_Library та Product_Category

Зображена на рис. 7.2 таблиця PRODUCT_LIBRARY містить інформацію про базову інформацію про тариф:

- id – первинний та зовнішній ключ таблиці;
- description – опис тарифу;
- duration – строк дії тарифу (може бути день, тиждень, місяць, рік);
- name – ім’я тарифу;
- price – ціна тарифу;

- product_category_id – зв'язок з категорією продукту – використовується зовнішній ключ таблиці PRODUCT_CATEGORY.

Зображена на рис. 7.2 таблиця SERVICE містить інформацію про можливі для підключення сервіси:

- id – первинний та зовнішній ключ таблиці;
- description – опис сервісу;
- duration – строк дії сервісу (може бути день, тиждень, місяць, рік);
- name – ім’я сервісу;
- price – ціна сервісу;
- product_category_id – зв'язок з категорією продукту – використовується зовнішній ключ таблиці PRODUCT_CATEGORY.

Зображена на рис. 7.2 таблиця PRODUCT містить інформацію про конкретний тарифний план підключений користувачем:

- id – первинний та зовнішній ключ таблиці;
- product_template_id – зв'язок з таблицею PRODUCT_LIBRARY за допомогою зовнішнього ключа цієї таблиці.

Також містить зв'язок багато до багатьох з таблицею SERVICES за допомогою таблиці PRODUCT_SERVICES, яка зв’язує ці дві таблиці за допомогою зовнішніх ключів.

Зв'язок між таблицями USER і PRODUCT та USER і SERVICE відбувається використовуючи допоміжні таблиці USER_PRODUCT і USER_SERVICE. Вони містять зовнішні ключи таблиць USER і PRODUCT та USER і SERVICE відповідно.

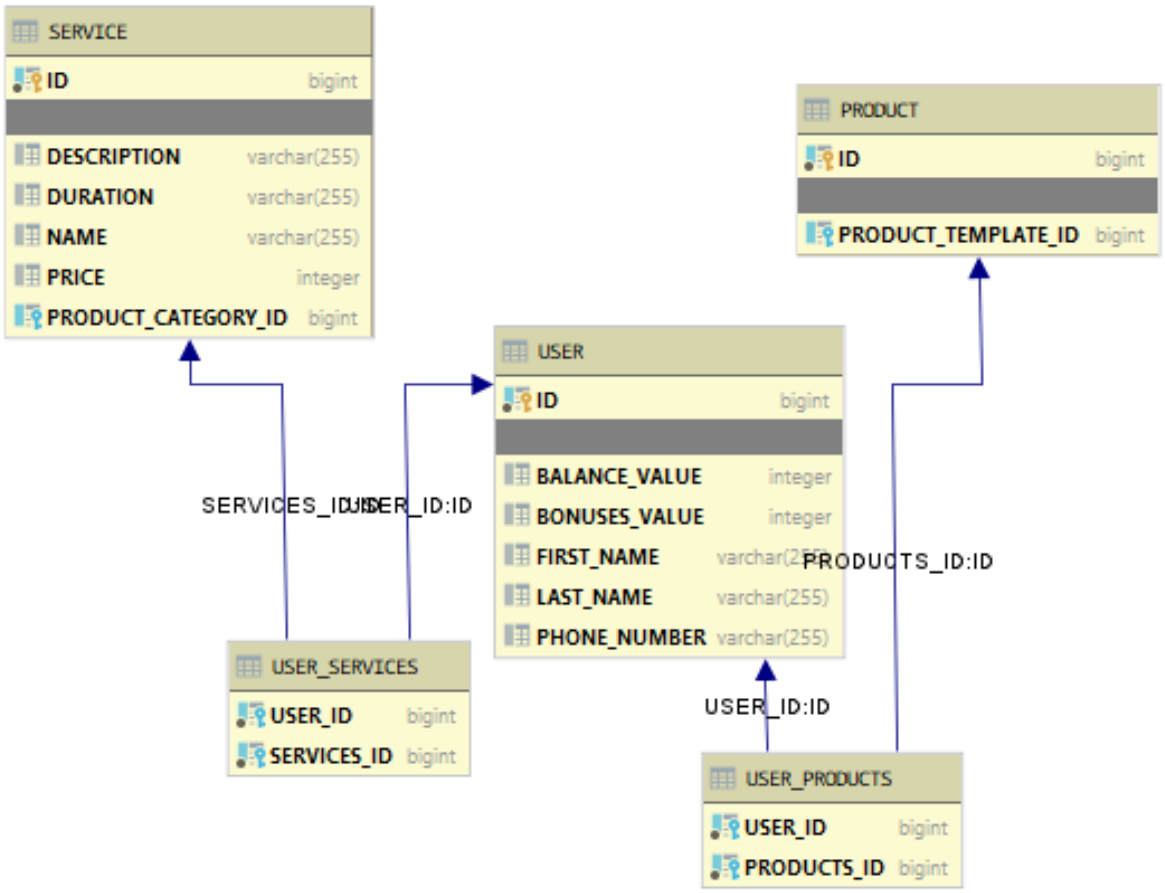


Рисунок 7.3 – Зв'язок між сутностями User з сутностями Product і Service

Зображена на рис.7.4 таблиця LOYALTY містить інформацію про пропозиції програми лояльності:

- id – первинний ключ таблиці;
- bonus_price – ціна підключення пропозиції в бонусах;
- description – опис пропозиції;
- name – назва пропозиції.

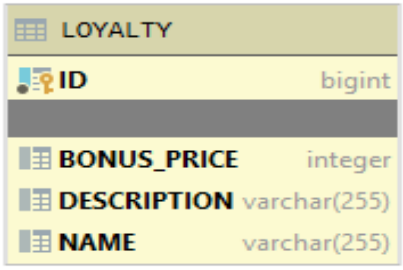


Рисунок 7.4 – Сутність Loyalty

Система для створення ботів містить 3 таблиці для роботи з користувачем та для зберігання інформації яка міститься в InlineKeyboardButton.

Зображена на рис.7.5 таблиця USER містить інформацію про користувача, який вже підписаний на використання бота:

- id – первинний ключ таблиці. Отримується з серверної частини CRM-системи та дорівнює унікальному ідентифікатору користувача у цій системі;
- chat_id – ідентифікатор чату бота з користувачем. Отримується за допомогою Telegram API;
- first_name – им’я користувача, отримується з серверної частини CRM-системи;
- last_name – прізвище користувача, отримується з серверної частини CRM-системи;
- phone_number – номер мобільного телефону користувача, отримується за допомогою Telegram API. За допомогою нього отримується сутність User серверної частини CRM-систем
- telegram_id – ідентифікатор користувача у месенджері Telegram, отримується за допомогою Telegram API. Є зовнішнім ключем таблиці.

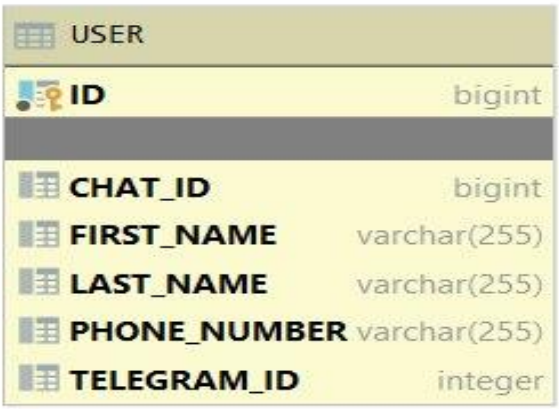


Рисунок 7.5 – Сутність User

Зображена на рис.7.6 таблиця ACTION_ITEM містить інформацію, що саме повинно відображатись в конкретний момент на InlineKeyboardButton:

- id – первинний та зовнішній ключ таблиці;
- action – url ресурсу серверної частини CRM-системи, на який потрібно звернутись при натисканні;
- message – текст кнопки;
- method – який HTTP метод потрібно використати при зверненні до серверної частини CRM-системи. Можливі значення GET, POST, PUT, DELETE.

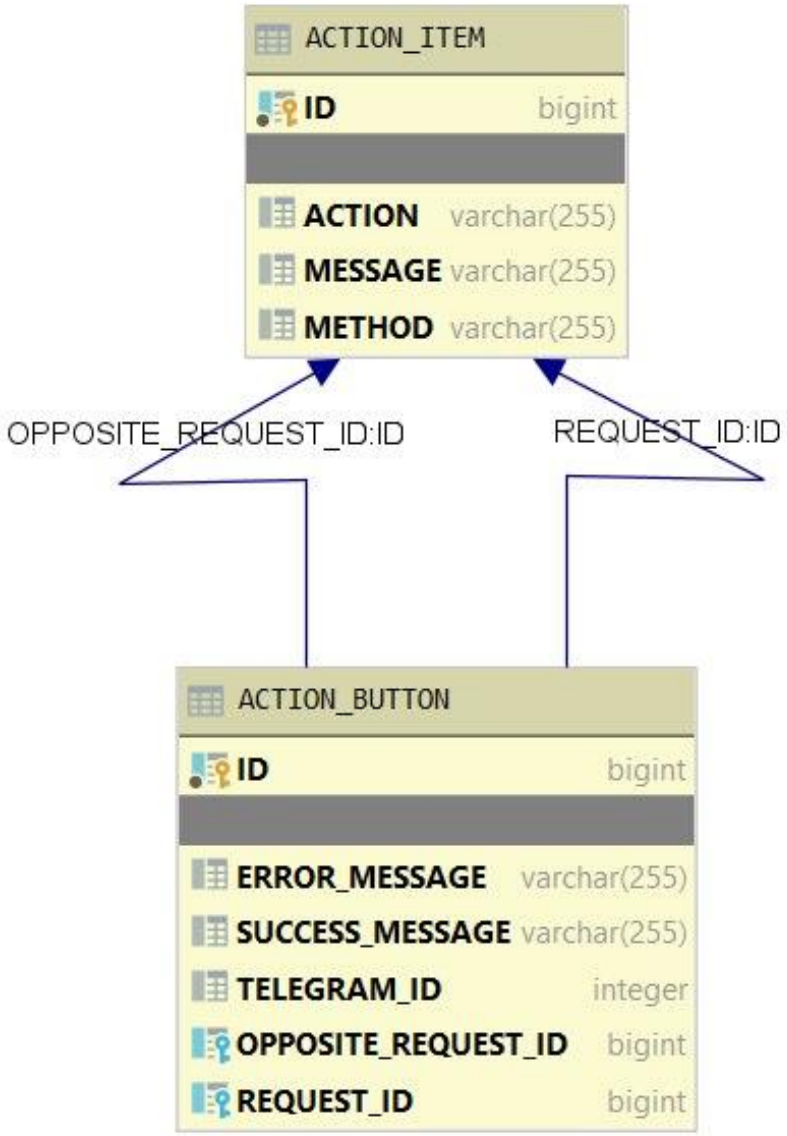


Рисунок 7.6 – Сутності ActionItem та ActionButton

Зображена на рис.7.6 таблиця ACTION_BUTTON містить повну інформацію про кнопку InlineKeyboardButton:

- id – первинний ключ таблиці;
- error_message – текст, який потрібно відобразити, якщо запит на серверну частину CRM-системи пройшов невдало;
- success_message – текст, який потрібно відобразити, якщо запит на серверну частину CRM-системи пройшов вдало;
- telegram_id – ідентифікатор користувача у месенджері Telegram, є зовнішнім ключем таблиці USER. Використовується для видалення всієї множини об’єктів ActionButton, пов’язаних з користувачем. Таким чином база даних не містить застарілу інформацію, яка не використовується;
- request_id – зовнішній ключ таблиці ACTION_ITEM. Містить зв’язок між загальною інформацією про кнопку та інформацію, яка відображається прямо зараз;
- opposite_request_id – зовнішній ключ таблиці ACTION_ITEM. Містить зв’язок між загальною інформацією про кнопку та інформацію, яка відображається у разі вдалого проходження операції при натисканні.

8 РЕАЛІЗАЦІЯ БІЗНЕС-ЛОГІКИ КОРИСТУВАЧА

Нижче буде наведено поведінку системи при виконанні основних операцій отримання та зміни інформації користувачем. Також у Додатку К зображено діаграму послідовності на прикладі перегляду та активації тарифного плану.

8.1 Авторизація користувача

1) Перевірка існування користувача у базі даних чат-боту.
2) Якщо ні, то послати запит на отримання користувача у серверну частину за номером телефону.
3) Серверна частина шукає користувача у своїй базі даних та повертає результат.
4) Якщо користувача не знайдено, то зображується повідомлення помилки;
5) Якщо знайдено, то користувач зберігається у базі даних бота.
6) Після цього з’являється головне меню.

8.2 Отримання списку тарифів, підключення та відключення тарифу

Передумова: користувач є зареєстрованим та підписаним на бота.

1) Користувач переходить на в пункт меню «Products».
2) Користувач обирає категорію.
3) Бот посилає запит на отримання списку тарифів у серверну частину.
4) Серверна частина отримує список тарифів з бази даних, проводить конвертацію моделей та надсилає боту.
5) Бот отримує список тарифів та заповнює повідомлення.
6) Бот заповнює інформацію для варіантів використання InlineKeyboardButton, зберігає їх у власну базу даних та створює

InlineKeyboardButton з цими значеннями та записує ідентифікатор в цю кнопку відповідного запису у БД.

7) Бот надсилає повідомлення користувачу зі списком всіх тарифів.
8) Користувач підключає/відключає тариф натискаючи InlineKeyboardButton.
9) Бот отримує ідентифікатор запису у БД та дістає його.
10) Бот надсилає запит на підключення/відключення у серверну частину.
11) Серверна частина валідує операцію та підключає/відключає тариф та надсилає код успішності/неуспішності операції.
12) Бот отримує статус відповіді та якщо операція неуспішна користувачеві відправляється відповідне повідомлення.
13) Якщо операція успішна, то бот проводить зміну у базі даних стосовно інформації про InlineKeyboardButton.
14) Бот корегує значення InlineKeyboardButton у повідомленнях.
15) Користувачеві надсилається повідомлення про успіх операції.

8.3 Отримання списку сервісів, підключення та відключення сервісу

Передумова: користувач є зареєстрованим та підписаним на бота.

1) Користувач переходить на в пункт меню «Services».
2) Бот посилає запит на отримання списку сервісів у серверну частину.
3) Серверна частина отримує список сервісів з бази даних, проводить конвертацію моделей та надсилає боту.
4) Бот отримує список сервісів та заповнює повідомлення.
5) Бот заповнює інформацію для варіантів використання InlineKeyboardButton, зберігає їх у власну базу даних та створює InlineKeyboardButton з цими значеннями та записує ідентифікатор в цю кнопку відповідного запису у БД.
6) Бот надсилає повідомлення користувачу зі списком всіх сервісів.

7) Користувач підключає/відключає сервіс натискаючи InlineKeyboardButton;
8) Бот отримує ідентифікатор запису у БД та дістає його.
9) Бот надсилає запит на підключення/відключення у серверну частину.
10) Серверна частина валідує операцію та підключає/відключає сервіс та надсилає код успішності/неуспішності операції.
11) Бот отримує статус відповіді та якщо операція неуспішна користувачеві відправляється відповідне повідомлення.
12) Якщо операція успішна, то бот проводить зміну у базі даних стосовно інформації про InlineKeyboardButton.
13) Бот коригує значення InlineKeyboardButton у повідомленнях.
14) Користувачеві надсилається повідомлення про успіх операції.

8.4 Отримання списку пропозицій програми лояльності, використання бонусів на придбання пропозиції

Передумова: користувач є зареєстрованим та підписаним на бота.

1) Користувач переходить на в пункт меню «Bonus Program».
2) Бот посилає запит на отримання списку пропозицій у серверну частину.
3) Серверна частина отримує список пропозицій з бази даних, проводить конвертацію моделей та надсилає боту.
4) Бот отримує список пропозицій та заповнює повідомлення.
5) Бот заповнює інформацію для варіантів використання InlineKeyboardButton, зберігає їх у власну базу даних та створює InlineKeyboardButton з цими значеннями та записує ідентифікатор в цю кнопку відповідного запису у базу даних.
6) Бот надсилає повідомлення користувачу зі списком всіх пропозицій.
7) Користувач підключає пропозицію натискаючи InlineKeyboardButton.
8) Бот отримує ідентифікатор запису у базу даних та дістає його.
9) Бот надсилає запит на підключення у серверну частину.

10) Серверна частина валідує операцію та підключає пропозицію та надсилає код успішності/неуспішності операції.
11) Бот отримує статус відповіді та якщо операція неуспішна користувачеві відправляється відповідне повідомлення.
12) Користувачеві надсилається повідомлення про успіх операції.

8.5 Перегляд інформації щодо підключених тарифах та сервісах

Передумова: користувач є зареєстрованим та підписаним на бота.

1) Користувач переходить в пункт меню «Settings».
2) Користувач переходить в пункт меню «My Products» чи «My Services».
3) Бот надсилає запит на отримання в серверну частину.
4) Серверна частина отримує список підключених тарифів/сервісів, проводить конвертацію моделей та надсилає список боту.
5) Бот отримує список тарифів/сервісів та заповнює повідомлення.
6) Бот відправляє повідомлення.

8.6 Перегляд інформації щодо отриманих пропозиціях програми лояльності

Передумова: користувач є зареєстрованим та підписаним на бота.

1) Користувач переходить в пункт меню «Settings».
2) Користувач переходить в пункт меню «My Bonuses».
3) Бот надсилає запит на отримання в серверну частину.
4) Серверна частина отримує список отриманих пропозицій, проводить конвертацію моделей та надсилає список боту.
5) Бот отримує список пропозицій та заповнює повідомлення.
6) Бот відправляє повідомлення.

8.7 Перегляд інформації щодо стану рахунку

Передумова: користувач є зареєстрованим та підписаним на бота.

1) Користувач переходить в пункт меню «Settings».
2) Користувач переходить в пункт меню «My Balance».
3) Бот надсилає запит на отримання в серверну частину.
4) Серверна частина отримує сутність користувача проводить конвертацію моделі та надсилає список боту.
5) Бот отримує стан рахунку та заповнює повідомлення.
6) Бот відправляє повідомлення.

9 НАЛАШТУВАННЯ ЧАТ-БОТУ

Основною ідеєю налаштування бота через окремий файл є те, що в такому випадку після внесення змін в налаштування достатньо буде перезапустити систему і не потрібно буде збирати заново всю систему.

Формат файлу було обрано yaml чи yml. У цьому рішенні є дві причини:

- існують зручні бібліотеки, за допомогою яких можна заповнювати поля об’єкту після старту системи. Таким чином одразу отримується звичайний Java об’єкт, з яким можна зручно працювати та впроваджувати в інші класи;
- це формат створений для зручного задання ієрархічної структури. Оскільки робота бота заснована на використанні меню, а меню в широкому сенсі має деревоподібну структуру, то цей формат файлу налаштувань є візуально зручним для заповнення.

Нижче наведено приклад вигляду файлу та будуть наведені пояснення для полів. Зміст файлу складається з двох частин, які зіставляються з двома різними Java об’єктами. Перша частина містить загальні налаштування бота:

```
pathToEnv: http://localhost:8087
userApiPath: /user/phone/{phone}
botUsername:
botToken:
userNotFoundMessage: You're not active user. You can sign out here – http://somelink.com
userExists: You're signed in successfully
```

Пояснення для чого потрібні ці поля:

- pathToEnv – загальний шлях до серверної частини CRM-системи;
- userApiPath – шлях, за яким бот може отримати сутність користувача системи з серверної частини CRM-системи. {phone} – це номер телефону користувача, який бот отримує за допомогою Telegram API. Результуючим шляхом буде pathToEnv + userApiPath;

- botUsername – ім’я бота, якого створено за допомогою центрального бота Телеграм – BotFather;
- botToken – токен бота, якого створено за допомогою центрального бота Телеграм – BotFather;
- userNotFoundMessage – повідомлення, яке має зобразитись коли користувач не зареєстрований в системі;
- userExists – повідомлення, яке зобружається якщо процес авторизації пройшов успішно.

Друга частина файлу з налаштуваннями містить інформацію про доступні пункти меню, як вони розташовані, та які дії потрібно виконати при переходячи на пункт меню:

```
menu:
 message: Welcome to main menu
 menuItems:
         – name: Services
         path: /services/{userId}
         message: >
                <emoji:exclamation:>[{category}] {name} |
                <emoji:white_check_mark:>{description}|
                Price: {price}UAH/{duration}
         action: <if {isActive} value='/user/{userId}/service/{id}' method='DELETE'><else
value='/user/{userId}/service/{id}' method='POST'>
         actionMessage: <if {isActive} value='Deactivate service'><else value='Activate
service'>
         text: It's our services
         successMessage: Successful!
         errorMessage: Something went wrong

         -name: Settings
          path: null
          message: There are information about your profile
          menuItems:
                 -name: My products
                  path: /user/{userId}/products
                  message: >
                <emoji:large_blue_circle:>[{productTemplate.productCategory.name}]
        {productTemplate.name} |
                {productTemplate.description}
```

```
-name: My services
 path: /user/{userId}/services
 message: >
      <emoji:red_circle:>[{productCategory.name}] {name} |
      {description}
```

Пояснення для чого потрібні ці поля:

- menu – кореневий елемент меню, саме по ньому зіставляється Java об’єкт та цей фрагмент файлу;
- message – повідомлення, яке зображується коли користувач знаходиться у корені меню (у випадку, якщо він вже зареєстрований);
- menuItems – масив пунктів меню, які знаходяться під кореневим елементом (така сама поведінка є і в пунктах меню – вони теж можуть містити масив пунктів-нащадків, як у пункті меню «Settings», так і не містити як, наприклад, в «Services»).

Далі буде розглянуто виключно поля пункту меню на прикладі пункту «Services» та «Settings»:

- name – ім‘я пункту меню – це значення буде відображатись на кнопці;
- path – шлях, по якому треба перейти при переході на пункт меню. Якщо в шляху є ідентифікатор користувача, то він повинен шаблонізуватись за допомогою {userId}. Результуючим шляхом буде pathToEnv + path;
- text – загальне повідомлення, яке буде зображатись першим. Це не обов’язкове поле і потрібно лише задля узагальнення списку повідомлень про однакові об’єкти;
- message – шаблон для повідомлення, яке буде зображатись для кожного об’єкта, отриманого при запиті при переході на цей пункт меню. Серед цікавих особливостей задання шаблону можна відмітити:

- {name} – означає, що замість нього треба підставити значення поля name з отриманого об’єкту. Якщо ж структура поля складна, то можна вказати шлях до поля внутрішнього об'єкта, як, наприклад, {productCategory.name};
- Для того, щоб розбити повідомлення на декілька рядків слугує знак «|». При обробці повідомлення цей знак заміняється переносом рядка;
- Повідомлення підтримує наявність стандартних емодзі. Для цього потрібно до шаблону повідомлення додати емодзі у форматі: <emoji:exclamation:>, де :exclamation: – це код стандартного емодзі. При обробці повідомлення вся структура зміниться на емодзі.

Наступні чотири поля не є обов’язковими та потрібні лише у випадку, коли потрібно під повідомлення додати кнопку:

- action – потрібно для отримання шляху та HTTP методу для при натисканні на кнопку. Містить два ключові поля method та value. У випадку існування протилежної операції, на кшталт підключити/відключити, значення поля можна задати за допомогою if-else конструкції. Для цього перевіряється чи дорівнюється «true» значення поля отриманого об’єкту. Якщо так, то береться перша частина, ні – друга;
- actionMessage – містить таку саму логіку як і для action та потрібна для знаходження тесту для кнопки;
- successMessage – текст повідомлення, яке виводиться, якщо операція після натискання кнопки була успішною;
- errorMessage – текст повідомлення, яке виводиться, якщо операція після натискання кнопки була неуспішною.

10 ІНТЕРФЕЙС КОРИСТУВАЧА

Оскільки завдяки інтеграції бота з месенджером Телеграм немає необхідності розробляти власний інтерфейс, то зовнішній вид бота виглядає стандартно. Але розглянемо як саме зображуються кнопки меню, як виглядають повідомлення використовуючи серверну частину CRM-системи телекомунікаційної компанії.

1. Тож, у випадку, коли користувач тільки-но підписався на використання бота бот запрошує поділитись номером мобільного телефону, як видно на рис.10.1.

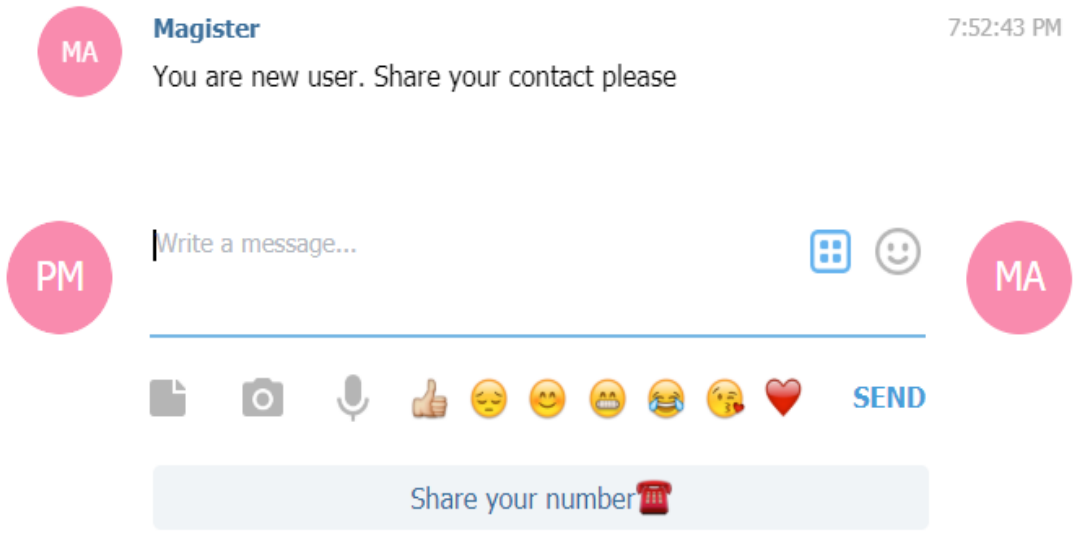


Рисунок 10.1 – Авторизація

2. Після успішної авторизації користувач потрапляє в головне меню, де може перейти до перегляду тарифів, сервісів, пропозицій програм лояльності та до переглядання інформації про підключені продукти, як видно на рис.10.2.

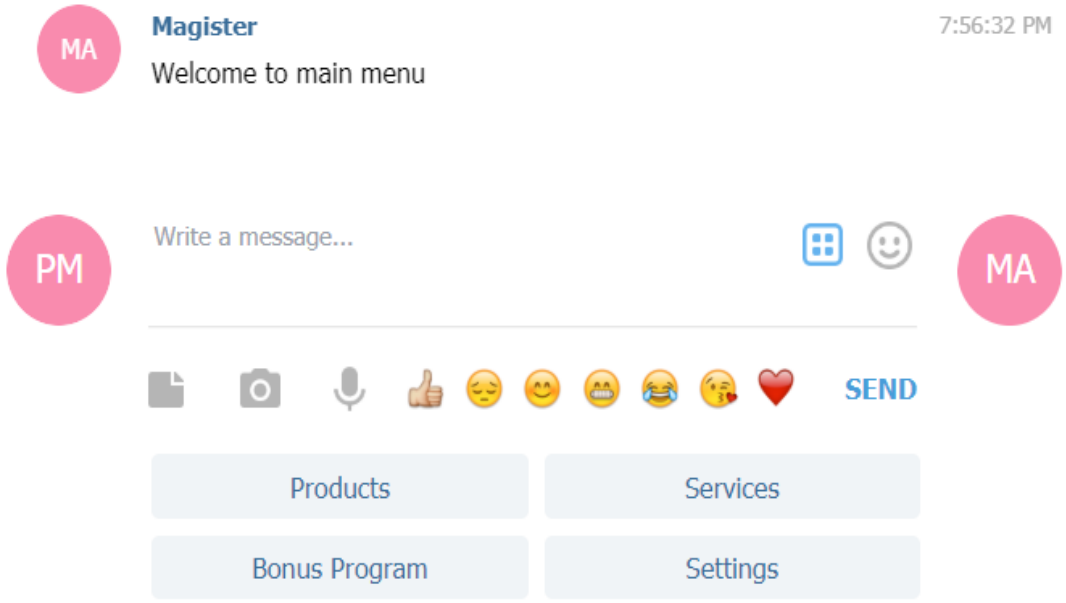


Рисунок 10.2 – головне меню

3. Після того, як користувач перейде до перегляду тарифів та вибере категорію, яка його цікавить він побачить список можливих товарів, при чому одразу видно які в нього є підключеними, а які ні. Також для зручної навігації по дереву пунктів меню в кожному пункті, окрім кореневого, є кнопка повернення на попередній рівень меню. Причому, якщо попередній рівень є кореневим, то кнопка має напис «Back», в іншому випадку – «Back to» і назва попереднього рівня. Ця кнопка є системною та не потребує додаткових налаштувань. Список тарифних планів можна побачити на рис.9.3.
4. Повернемось до кореневого меню та перейдемо до перегляду сервісів. Логіка підключення та відключення сервісу така сама як і у тарифного плану. Список сервісів можна побачити на рис.10.4.

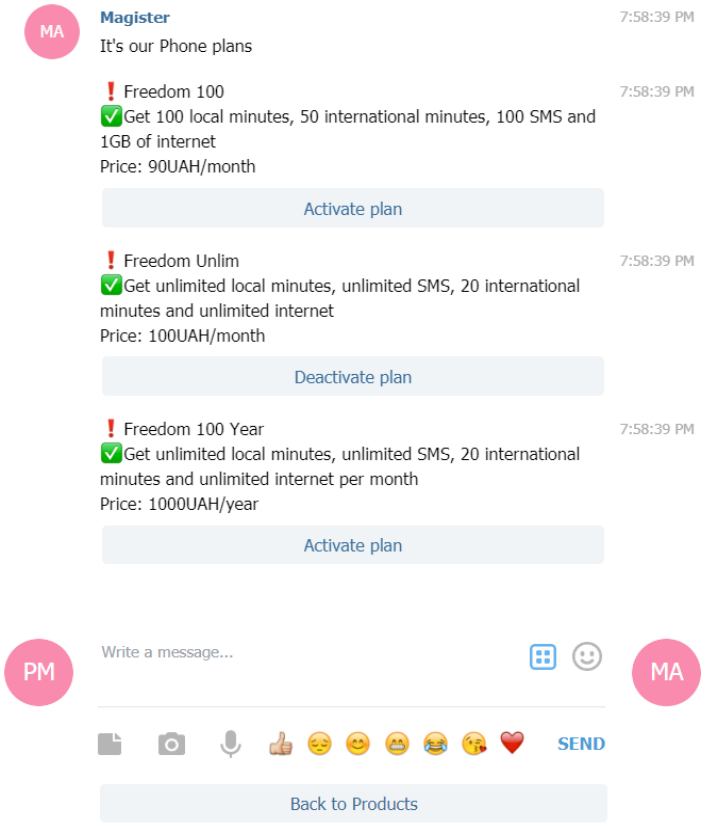


Рисунок 10.3 – тарифні плани

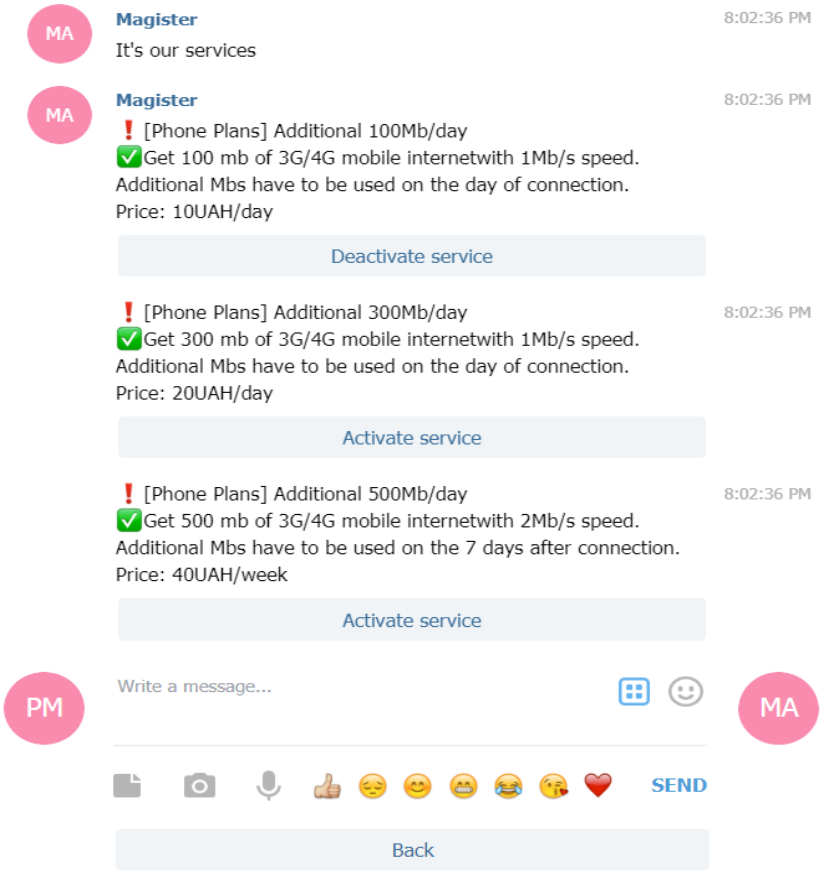


Рисунок 10.4 – список сервісів

5. Повернувшись до кореневого меню користувач може переглянути пропозиції програми лояльності та підключити вподобану пропозицію. Як це виглядає можна побачити на рис.9.5.

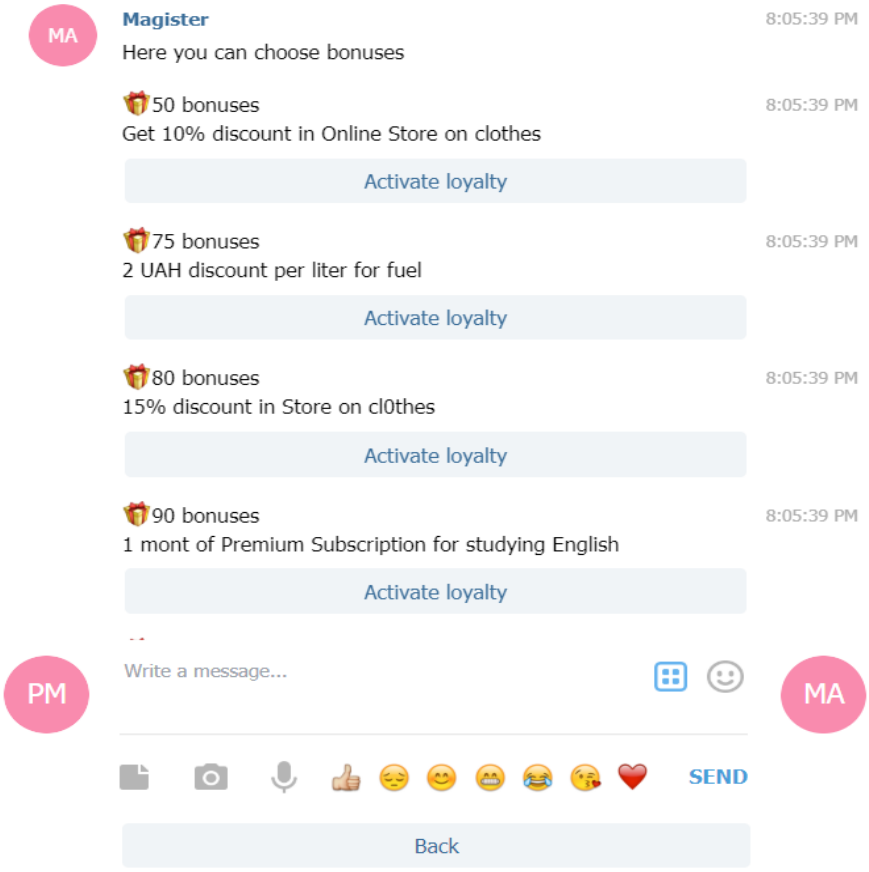


Рисунок 10.5 – список пропозицій бонусної програми

6. У меню налаштувань користувач може переглянути під’єднані тарифи та сервіси. Для цього потрібно спочатку перейти в пункт меню «Settings», а потім перейти до пунктів меню «My Products» чи «My Services» відповідно. Також в кожному елементі списків також зображується назва категорії продукту. Побачити це можна на рис.10.6 та рис.10.7.

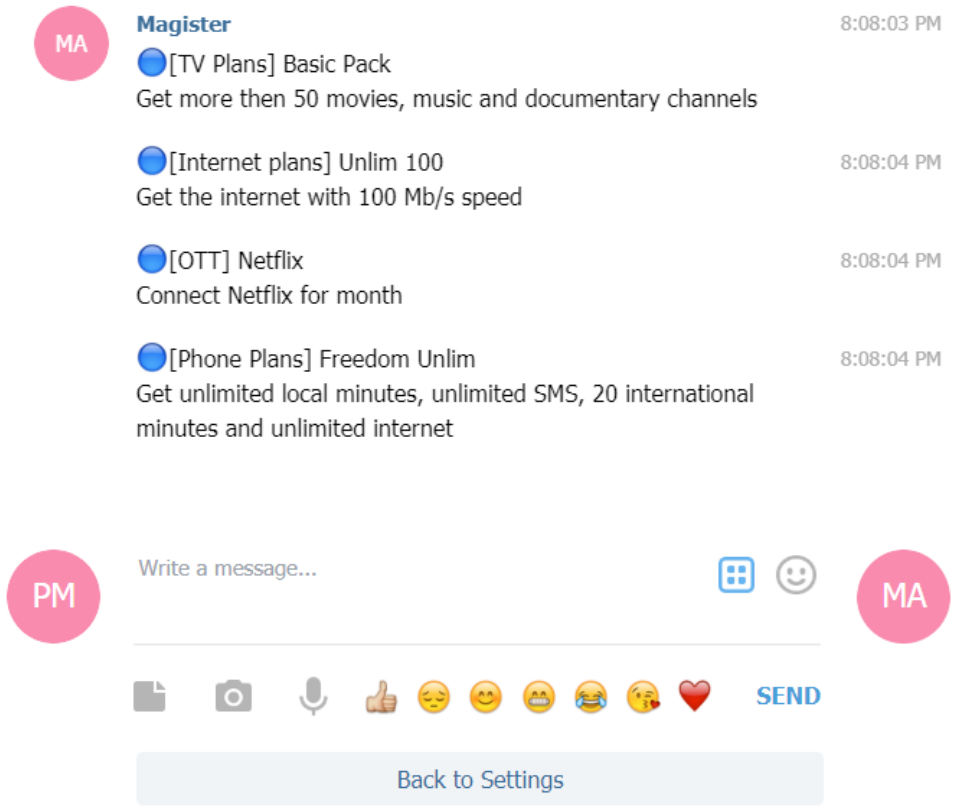


Рисунок 10.6 – Список підключених тарифів

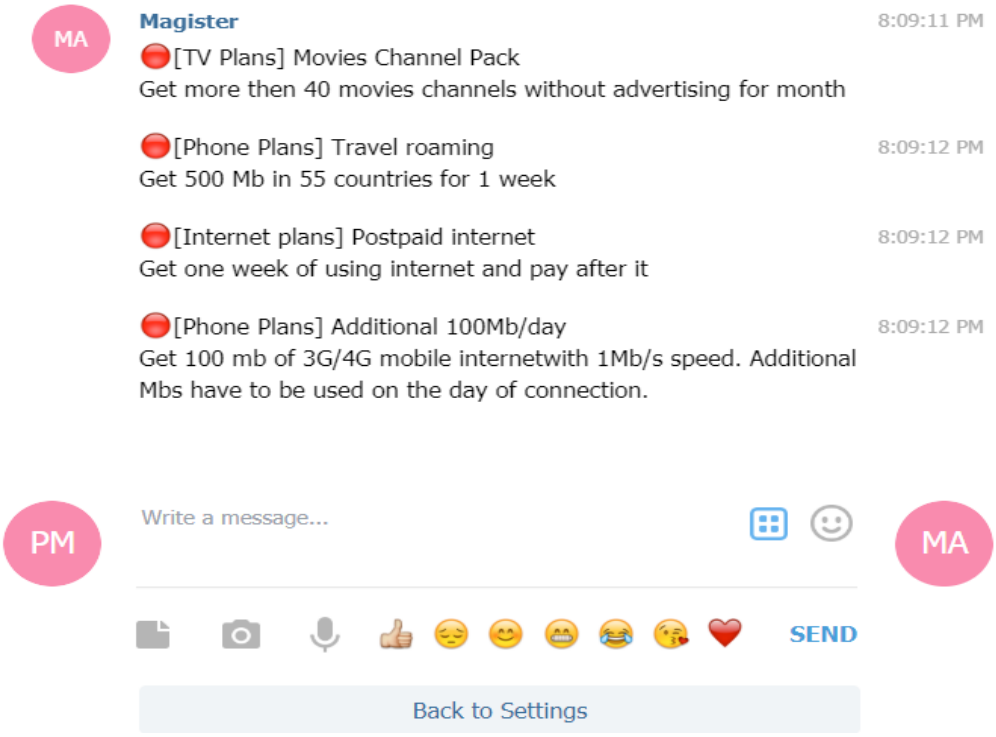


Рисунок 10.7 – Список підключених сервісів

7. Також можна переглянути усі отримані бонуси перейшовши в налаштуваннях в пункт меню «My bonuses». Як це виглядає можна побачити на рис.10.8.

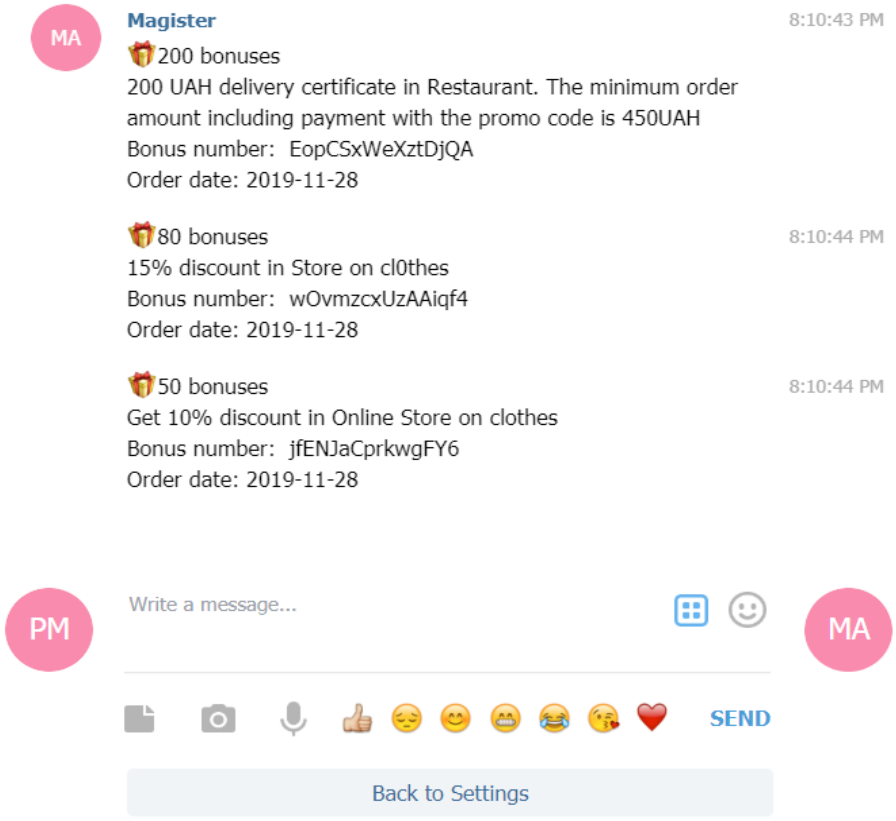


Рисунок 10.8 – список отриманих пропозицій

6.2.7 А також подивитися стан свого рахунку перейшовши в пункт меню «My Balance».Побачити це можна на рис.10.9.

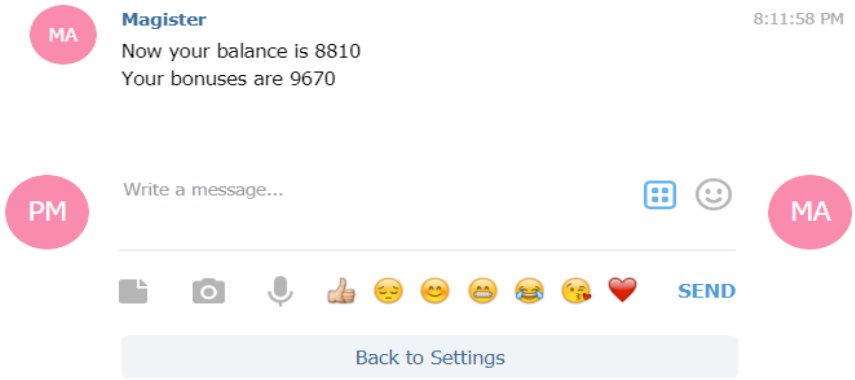


Рисунок 10.9 – стан рахунку

9. Також бот може відправляти повідомлення всім підписаним на нього користувачам або конкретному користувачу. Це виглядає можна побачити на рис.10.10.

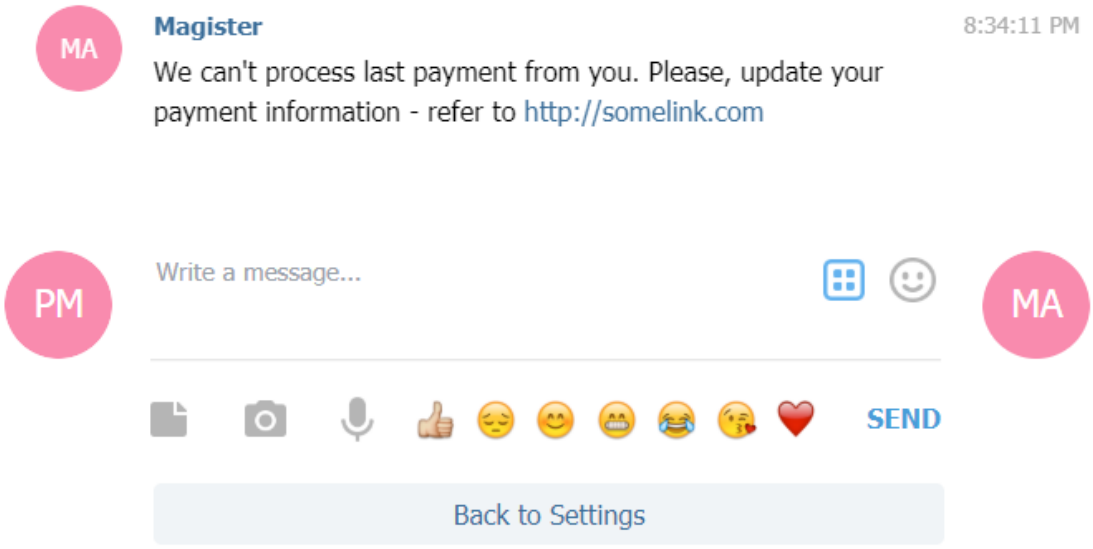


Рисунок 10.10 – отримане повідомлення

Висновком може бути, що бот правильно опрацьовує свої налаштування, все зображується саме так, як було заплановано користувачем: показуються всі пункти меню, їх ієрархія коректна, повідомлення заповнюються даними, які прийшли з серверної частини, емодзі правильно показуються, саме повідомлення розбивається на рядки. Завдяки системній кнопці повернення на минулий рівень меню відбувається зручна навігація. А отже простими налаштуваннями було отримано зручного та інтуїтивно зрозумілого бота.

11 ТЕСТУВАННЯ СИСТЕМИ

Тестування програмного забезпечення це невід’ємний процес розробки програмного забезпечення. Він переслідує одразу декілька цілей.

1) Оцінка якості продукту. В це поняття входить як процес пошуку дефектів та помилок так і оцінка швидкодії системи, її безпека та використання на різних платформах.
2) Відповідність функціональним вимогам. В це поняття входить оцінка того, що написана система відповідає задумці, виконує всі заплановані дії та коректно працює в межах заданих дій.
3) Запобігання дефектів. В майбутньому система може розвиватись, а отже змінюватись. У цьому випадку доволі частим є явище зміни вже написаного коду у результаті зміни підходу. Для цього треба бути впевненим в тому, що зміни в одній частині проекту не торкнуться інших частин.

Виходячи з цього існують різні види тестування, кожний з яких є підґрунтям для виявлення якості продукту. Отже, за ступенем ізольованості існують.

1) Модульне тестування – оцінюються окремі компоненти системи незалежно один від одного.
2) Інтеграційне тестування – тестування взаємодії різних компонентів між собою.
3) Системне тестування – тестування інтегрованої системи на відповідність всім вимогам.

В залежності від переслідуваних цілей існують такі види тестування.

1) Функціональне тестування – тестування відповідності функціональним вимогам.

2) Нефункціональне тестування – тестування виконання нефункціональним вимогам.
3) Регресивне тестування – це тестування пов’язане зі змінами. Цей вид спрямований на пошук та виявлення помилок у вже протестованих частинах коду. Зазвичай автоматично виконуються у CI/CD для великих за об’ємом та часом проектів.

За об’єктом тестування існують такі основні різновиди тестування.

1) Функціональне тестування.
2) Тестування продуктивності та навантажувальне тестування – основна ціль цього типу це оцінка продуктивності продукту та дослідження його ефективності в умовах підвищеного навантаження (імітування великої кількості одночасно працюючих користувачів, велика кількість транзакції виконаних одночасно, тощо). Оскільки метою цих видів тестування є оцінка продуктивності, то воно не розділені.
3) Тестування безпеки – це тестування як створених компонентів системи так і використаних у розробці на виявлення дефектів у безпеці продукту. Результатом цього виду тестування є список вразливостей за допомогою яких зловмисники можуть отримати несанкціонований доступ до даних.

В рамках цього розділу буде розглянуто модульне та функціональне тестування створеного продукту.

11.1 Модульне тестування

Модульне тестування чи юніт тестування (від слова англійського unit – одиниця, частина, елемент) – це перевірка, що окремі компоненти системи працюють саме так, як задумано розробниками. В залежності від типу розробки визначається коли саме будуть створенні тести. І якщо у випадку поетапного процесу розробки написання тестів починається після визначення

системних вимог, то у випадку гнучкого підходу до створення програмного забезпечення створення тестів та написання тестів може відбуватись одночасно.

Треба зазначити, що роль модульного тестування в гнучких методах розробки програмного забезпечення значно відрізняється від більш традиційних методологій. Якщо у традиційних методах тести пишуться для знаходження дефектів, то їх створення в контексті розробки, заснованої на тестуванні в першу чергу слугує для довготривалої підтримки та контролю створеного функціоналу. Під час розширення функціоналу системи чи її оновлення зміни в одній частині системи можуть торкнутися інших компонентів, що може призвести до появи нових дефектів у вже протестованій компоненті. Також не можна обійти питання рефакторінгу, який є природним етапом процесу розробки програмного забезпечення.

Рефакторінг – це перетворення програмного коду, зміна внутрішньої структури програмного забезпечення чи декомпозиція системи, яка не впливає на функціонал системи. Процес створення розробки продукту не буває лінійним, тобто не можлива ситуація коли код написано, відтестовано і більше не змінено. Навіть якщо система якісно спроектована та всі поставленні вимоги дотримані, то все-одно будуть зміни. Це стосується не тільки планової підтримки системи, а й постійних правок до вимог. Таким чином якісно створений програмний код може бути змінено задля дотримання нових завдань до системи. Також треба зазначити, що на відміну від більш традиційних методів розробки, які ставлять акцент на мінімізації змін до коду, гнучкі методи розробки програмного забезпечення вбачають великий потенціал в змінах. Оскільки вони є більш «код-орієнтованими», то процес розробки включає в себе постійний рефакторінг та покращення створеного програмного коду.

Оскільки процес рефакторінгу може торкнутися будь-якої система треба бути впевненим, що функціональність системи залишається постійною. Також не потрібно забувати, що програмне забезпечення створюють багато

людей, розробка може тривати роками і в деякий час може наступити момент, коли навіть досвідчений розробник, який довго працює над системою не буде розуміти що саме можуть зачепити його зміни. Саме в цьому може допомогти модульне тестування. Оскільки тести пишуться одночасного зі створенням функціоналу, то на момент написання вони фіксують момент, коли написаний код працює правильно відносно задумки. Таким чином при випадковій зміні відразу буде видно місце, яке тепер функціонує неправильно.

Для модульного тестування системи було обрано бібліотеку JUnit 5. Створена Кентом Беком та Еріком Гаммою вона є представником родини фреймворків xUnit для різних мов програмування. Також було використано фреймворк Mockito для створення моків.

Оскільки в проектах класи залежать один від одного, то у модульному тестуванні часто виникає необхідність «підміняти» деякі результати, які повертають методи, чи передавати в метод деякі вручну сконфігуровані об’єкти. Це робиться для того, щоб протестувати саме цей об’єкт, правильність виконання його методів, а не тестувати роботу системи в цілому. Для цього існують деякі різновиди «підміни результатів»:

- dummy – це об’єкт, який часто передається в тестований клас в якості параметра. Але він не має поведінки, з ним нічого не відбувається. Доволі часто це є або null значення, або новий порожній об’єкт;
- stub використовується для отримання даних з якоїсь зовнішньої залежності чи компоненту. Іншими словами він заміняє її. При цьому абсолютно не звертається увага на передані параметри – відповідь завжди буде одна і та сама;
- spy використовується для тестів взаємодії. Дозволяє перевірити логіку тестованого об’єкта не перевіряючи коректність роботи залежних об’єктів;
- mock використовується для перевірки коректності поведінки тестованого об’єкта;

- fake – використовується для пришвидшення проходження тесту. Для цього «важка» зовнішня залежність підміняється більш легкою штучною реалізацією.

В рамках тестування створеної системи було протестована частину конструктора для чат-ботів. Розглянемо, для прикладу, тестовий клас для класу InlineButtonParser. Цей клас потрібен для створення з конфігураційного файлу InlineKeyboardMarkup об’єктів – активні кнопки які знаходяться під повідомленням від боту та можуть виконувати запит без явної команди та повідомлення. Розглянемо, яким чином відбувається різні види підміни результату.

Прикладом dummy:

```
String actual = inlineButtonParser.getActionMessage(TAG, getSimpleJsonNode(), true);
```

Як видно, логічна змінна true передається лише для перевірки роботи методу та більше ні на що не впливає.

Прикладом mock об’єкту може слугувати таке:

```
@Mock
private ActionButtonManager actionButtonManagerMock;
```

Тут звичайний ActionButtonManager обєкт, який є залежністю InlineButtonParser підміняється моком.

Прикладом stub можна розглянути таких фрагмент коду:

```
when(actionButtonManagerMock.saveActionButton(any(ActionButton.class)))
.thenReturn(8L);
```

За допомогою моку actionButtonManagerMock відтепер запит на збереження екземпляру інформації по кнопці, необхідної для створення InlineKeyboardButton нічого не зберігає, а лише повертає поле id об’єкту в базу даних. При чому абсолютно неважливо що саме передається в якості параметру методу.

Також у тестуванні є поняття позитивного та негативного тестування. Позитивне тестування – це написання такого модульного тесту, що відповідає ідеальним умовам, а отже використовуються такі тестові сценарії, що відповідають нормальній та очікуваній поведінці системи. На противагу,

негативне – це написання тестів з використання сценаріїв, які викликаю нештатну поведінку системи. Прикладом цього можуть слугувати два тесту, які тестують процес пошуку шаблонізованих змінних у заданій стрічці.

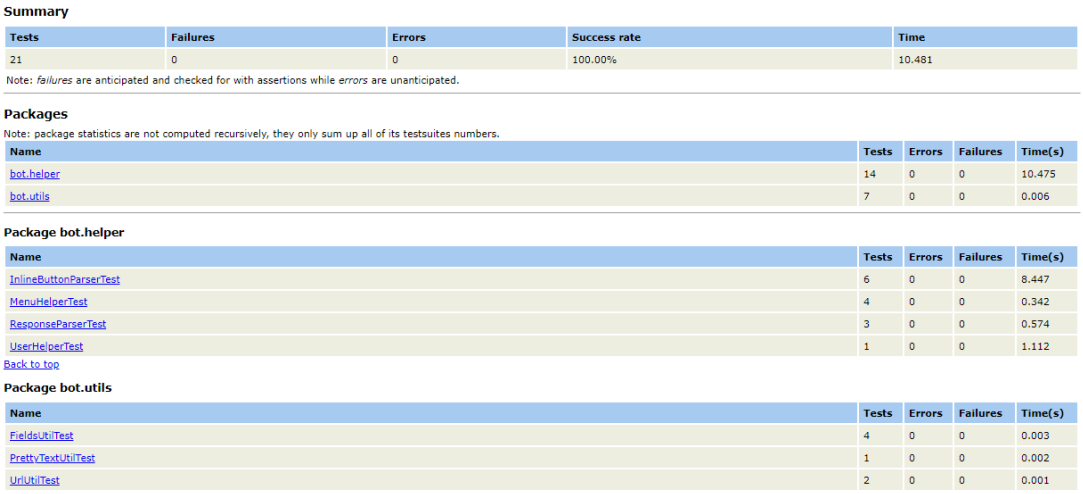
```
@Test
public void getFieldNamePatternsFromText() {
 List<String> actual = FieldsUtil.getFieldNamePatternsFromText(CORRECT_MESSAGE);
 List<String> expected = getFields();
 assertEquals(expected, actual);
}

@Test
public void getFieldNamePatternsFromWrongText(){
 List<String> fields = FieldsUtil.getFieldNamePatternsFromText(WRONG_MESSAGE);
 assertNull(fields);
}
```

Якщо перший тест є прикладом позитивного тестування, то другий тестує випадок, коли змінних немає чи вони задані неправильно.

Отже, всі необхідні тести написані і їх результат можна легко подивитись запустивши кожен. Проте, як вже вище згадано, тести потрібні не лише для перевірки якості коду та пошуку дефектів, а і для довготривалої підтримки створеного і протестованого програмного коду. Таким чином, необхідно, щоб запуск всіх тестів можна було зробити одночасно та автоматизовано. Оскільки для збирання проекту використовувався Maven, то його можна налаштувати таким чином, щоб при збиранні проекту також виконувались всі наявні модульні тести. Якщо ж якійсь тест не виконується, то процес створення збірки припиняється і показується тест, який не пройшов, що мотивує розробників ретельно все перевірити та виправити. Також можна зробити так, щоб результат проходження тестів зображався централізовано. рис.11.1 демонструє результати запуску тестів.

Також при оцінці якості тестування системи буває корисно знати яка саме кількість коду покрита тестами. Для початку розберемся, що таке покриття коду.



Summary

Tests	Failures	Errors	Success rate	Time
21	0	0	100.00%	10.481

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

Packages

Note: package statistics are not computed recursively, they only sum up all of its testsuites numbers.

Name	Tests	Errors	Failures	Time(s)
bot.helper	14	0	0	10.475
bot.utils	7	0	0	0.006

Package bot.helper

Name	Tests	Errors	Failures	Time(s)
InlineButtonParserTest	6	0	0	8.447
MenuHelperTest	4	0	0	0.342
ResponseParserTest	3	0	0	0.574
UserHelperTest	1	0	0	1.112

Back to top

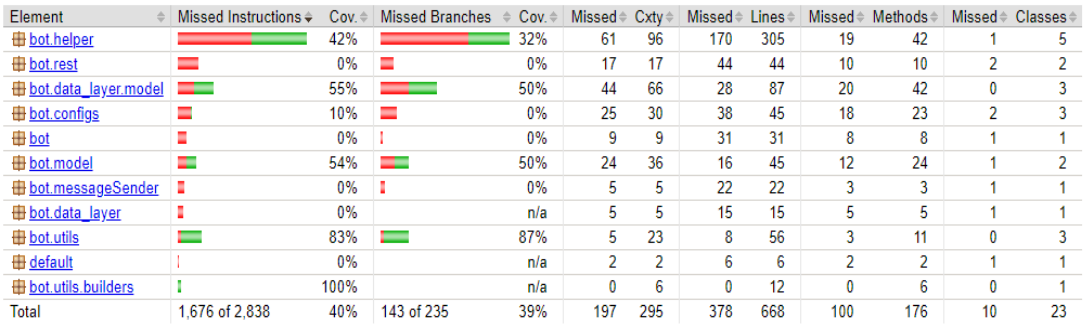
Package bot.utils

Name	Tests	Errors	Failures	Time(s)
FieldsUtilTest	4	0	0	0.003
PrettyTextUtilTest	1	0	0	0.002
UrlUtilTest	2	0	0	0.001

Рисунок 11.1 – Результат тестів

Покриття коду (Code coverage) – це деяка величина, яка відображає процент викликаного коду, який був викликаний при виконанні тестів. Значення цього проценту може коливатись від 0 до 100.

Для оцінки була використана бібліотека JaCoCo – бібліотека, яка допомагає в аналізі та зображенні оцінки покриття коду при тестуванні. На рис.11.2 зображено результат оцінки в даній системі.



Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
bot.helper		42%		32%	61	96	170	305	19	42	1	5
bot.rest		0%		0%	17	17	44	44	10	10	2	2
bot.data_layer.model		55%		50%	44	66	28	87	20	42	0	3
bot.configs		10%		0%	25	30	38	45	18	23	2	3
bot		0%		0%	9	9	31	31	8	8	1	1
bot.model		54%		50%	24	36	16	45	12	24	1	2
bot.messageSender		0%		0%	5	5	22	22	3	3	1	1
bot.data_layer		0%		n/a	5	5	15	15	5	5	1	1
bot.utils		83%		87%	5	23	8	56	3	11	0	3
default		0%		n/a	2	2	6	6	2	2	1	1
bot.utils.builders		100%		n/a	0	6	0	12	0	6	0	1
Total	1,676 of 2,838	40%	143 of 235	39%	197	295	378	668	100	176	10	23

Рисунок 11.2 – Оцінка покриття тестами

Отже, загальним результатом для всієї системи є значення покриття коду 40%, що є досить непоганим результатом, оскільки не тестувались елементарні методи, та методи, потрібні для виконання методів з зовнішніх залежностей.

11.2 Функціональне тестування

Функціональне тестування – це тестування відповідності роботи системи до того, як вона запланована працювати. Для цього пишуться тест-кейси, які описують пункти функціональних вимог та за допомогою них проводиться порівняння очікуваної поведінки та наявної. Отже, на основі функціональних вимог пишуться тест-кейси, які повинні повністю покрити їх всі.

Створимо для системи список тест-кейсів поданих у вигляді таблиць, які покривають функціональні вимоги.

Таблиця 11.1

Назва	Авторизація користувача
Номер	1
Опис	Дозволяє користувачу авторизуватись в системі за допомогою номеру телефону
Передумова	Користувач підписаний на використання чат-бота
Актор	Користувач
Головний сценарій	Користувач натискає кнопку «Share your number», після цього з’являється віконце від Telegram на згоду поділитися номером телефону. Після успішної авторизації з’являється повідомлення та меню чат-боту.
Альтернативний сценарій	У випадку відсутності користувача з таким номером телефону з’являється повідомлення про неможливість авторизації та посилання на реєстрацію в CRM-системі

Таблиця 11.2

Назва	Підключення та відключення тарифів
Номер	3
Опис	Дозволяє користувачу підключати та відключати тарифи
Передумова	Користувач підписаний на використання чат-бота та авторизований в ньому. Користувач знаходиться в корені меню
Актор	Користувач
Головний сценарій	Користувач натискає кнопку «Products», після цього з’являється йому пропонується вибрати категорію тарифу з таких: Phone plans, TV, Internet, OTT. При натискання відповідної кнопки з’являється список всіх доступний тарифів. Під кожним тарифом наявна кнопка з написом «Deactivate product» у випадку якщо тариф підключений та «Activate product» – у випадку якщо продукт не підключено. При натисканні продукт активується/деактивується і значення кнопки змінюється на протилежне.
Альтернативний сценарій	

Таблиця 11.3

Назва	Підключення та відключення сервісів
Номер	3
Опис	Дозволяє користувачу підключати та відключати сервіси

Передумова	Користувач підписаний на використання чат-бота та авторизований в ньому. Користувач знаходиться в корені меню
Актор	Користувач
Головний сценарій	Користувач натискає кнопку «Services», після цього з’являється список наявних сервісів. Під кожним сервісом наявна кнопка з написом «Deactivate Service» у випадку якщо сервіс підключений та «Activate» – у випадку якщо сервіс не підключено. При натисканні сервіс активується/деактивується і значення кнопки змінюється на протилежне.
Альтернативний сценарій	

Таблиця 11.4

Назва	Використання бонусів
Номер	4
Опис	Дозволяє користувачу користуватись бонусною програмою
Передумова	Користувач підписаний на використання чат-бота та авторизований в ньому. Користувач знаходиться в корені меню
Актор	Користувач
Головний сценарій	Користувач натискає кнопку «Bonus Program», після цього з’являється список пропозицій програми лояльності. Під кожною пропозицією наявна кнопка з написом «Activate loyalty». При натисканні пропозиція активується
Альтернативний сценарій	

Таблиця 11.5.1

Назва	Переглядання списку підключених тарифів
Номер	5
Опис	Дозволяє користувачу переглянути список підключених тарифів та інформацію по ним
Передумова	Користувач підписаний на використання чат-бота та авторизований в ньому. Користувач знаходиться в корені меню
Актор	Користувач
Головний сценарій	Користувач натискає кнопку «Settings», після цього з’являється меню власного кабінету. Користувач натискає кнопку «Services». З’являється список підключений сервісів.
Альтернативний сценарій	

Таблиця 11.5.2

Назва	Переглядання списку підключених сервісів
Номер	6
Опис	Дозволяє користувачу переглянути список підключених сервісів та інформацію по ним
Передумова	Користувач підписаний на використання чат-бота та авторизований в ньому. Користувач знаходиться в корені меню
Актор	Користувач
Головний сценарій	Користувач натискає кнопку «Settings», після цього з’являється меню власного кабінету. Користувач натискає кнопку «Products». З’являється список підключений тарифів.
Альтернативний сценарій	

Таблиця 11.6

Назва	Переглядання списку отриманих пропозиції програми лояльності
Номер	7
Опис	Дозволяє користувачу переглядати отримані бонуси та інформацію по ним
Передумова	Користувач підписаний на використання чат-бота та авторизований в ньому. Користувач знаходиться в корені меню
Актор	Користувач
Головний сценарій	Користувач натискає кнопку «Settings», після цього з’являється меню власного кабінету. Користувач натискає кнопку «My Bonuses». З’являється список підключений отриманих пропозицій бонусної програми.
Альтернативний сценарій	

Таблиця 11.7

Назва	Переглядання балансу користувача
Номер	8
Опис	Дозволяє користувачу користуватись подивитись стан свого рахунку та кількість бонусів у наявності
Передумова	Користувач підписаний на використання чат-бота та авторизований в ньому. Користувач знаходиться в корені меню
Актор	Користувач
Головний сценарій	Користувач натискає кнопку «Settings», після цього з’являється меню власного кабінету.

	Користувач натискає кнопку «My Balance». З’являється повідомлення зі станом балансу.
Альтернативний сценарій	

Таблиця 11.8

Назва	Переглядання балансу користувача
Номер	9
Опис	Відправлення повідомлення користувачеві
Передумова	Користувач підписаний на використання чат-бота та авторизований в ньому
Актор	Чат-бот
Головний сценарій	Система отримує запит ззовні який містить ідентифікатор користувача в системі та текст повідомлення та переправляє його в Telegram
Альтернативний сценарій	

11.3 Таблиця відповідності функціональних вимог

Після написання тест-кейсів функціональності системи створюється таблиця відповідності функціональних вимог (Traceability matrix) в якій ця відповідність фіксується.

Таблиця відповідності функціональних вимог – це двовимірна таблиця яка містить відповідність функціональних вимог та підготовлених тестових сценаріїв. У заголовку колонок таблиці записуються вимоги, а в заголовках рядків – тестові сценарії. На перетині колонки та рядка ставиться відмітка, яка показую що конкретний сценарії покриває вимогу. Таблиця є невід’ємною частиною тестування та використовується QA інженерами для перевірки покриття системи тестами.

В нас є 8 функціональних вимог, та 9 тестових сценаріїв. Побудуємо для них матрицю (таб.11.9).

Таблиця 11.9

Номер вимоги	1	2	3	4	5	6	7	8
Номер тестового сценарію								
1	+							
2		+						
3			+					
4				+				
5					+			
6					+			
7						+		
8							+	
9								+

Отже, всі функціональні вимоги повністю покриті тестовими сценаріями, а отже система побудована з урахуванням функціональних вимог.

12 СТАРТАП-ПРОЕКТ

Стартап як форма малого ризикового (венчурного) підприємництва впродовж останнього десятиліття набула широкого розповсюдження у світі через зниження бар'єрів входу в ринок (з появою Інтернету як інструменту комунікацій та збуту стало простіше знаходити споживачів та інвесторів, займатись пошуком ресурсів, перетинати кордони між ринками різних країн), і вважається однією із наріжних складових інноваційної економіки, оскільки за рахунок мобільності, гнучкості та великої кількості стартап-проектів загальна маса інноваційних ідей зростає.

Проте створення та ринкове впровадження стартап-проектів відзначається підвищеною мірою ризику, ринково успішними стає лише невелика частка, що за різними оцінками складає від 10% до 20%. Ідея стартап-проекту, взята окремо, не вартує майже нічого: головним завданням керівника проекту на початковому етапі його існування є перетворення ідеї проекту у працюючу бізнес-модель, що починається із формування концепції товару (послуги) для визначеної клієнтської групи за наявних ринкових умов.

Розроблення та виведення стартап-проекту на ринок передбачає здійснення низки кроків, в межах яких визначають ринкові перспективи проекту, графік та принципи організації виробництва, фінансовий аналіз та аналіз ризиків і заходи з просування пропозиції для інвесторів. Організацію стартап-проекту можна поділити на такі етапи

1. Маркетинговий аналіз стартап-проекту

   На цьому етапі:

   - розробляється опис ідеї продукту та визначаються його техніко-економічні характеристики;
   - визначаються його сильні та слабкі сторони у порівнянні з конкурентами;
   - аналізуються ринкові можливості щодо реалізації продукту;

- на базі аналізу ринкового середовища розробляється стратегія виводу на ринок потенційного продукту

2. Організація стартап-проекту

   На цьому етапі:

   - складається календарний графік реалізації стартап-проекту;
   - проводиться розрахунок матеріальних та нематеріальних активів;
   - виробляється плановий обсяг виробництва, для передбачення необхідності у матеріальних ресурсах та персоналі;
   - розрахунок витрат, необхідних для запуску стартапу та витрати, необхідні для реалізації продукту.

3. Фінансово-економічний аналіз та оцінка ризиків проекту

   На цьому етапі:

   - визначається загальний обсяг інвестицій;
   - розраховуються такі фінансово-економічні характеристики продукту як: обсяг виробництва продукції, собівартість виробництва, ціна реалізації, податкове навантаження та чистий прибуток;
   - визначаються такі характеристики інвестиційної привабливості продукту як: запас фінансової міцності, рентабельність продажів та інвестицій, період окупності проекту;
   - проводиться аналіз ризиків стартап-проекту та шляхи їх вирішення.

4. Дії, спрямовані на пошук потенційних інвесторів і створення та просування інвестиційної пропозиції.

   На цьому етапі:

   - визначається коло потенційних інвесторів;

- проводиться аналіз їх ділової зацікавленості;
- складання стислої характеристики проекту для попереднього ознайомлення інвестора із проектом;
- визначення комунікаційних каналів та площадок та планування системи заходів з просування в межах обраних каналів;
- плануються необхідні ресурси для просування оферти
- планування ресурсів для реалізації заходів з просування оферти.

Ці етапи, реалізовані послідовно та правильно, є передумовою успішного старту стартап-проекту.

12.1 Опис ідеї проекту

Для реалізації пункту «Маркетинговий аналіз стартап-проекту» було проаналізовано та подано у виді таблиці такі характеристики як:

- зміст ідей;
- напрямки застосування;
- основна користь для користувача;
- відмінність від існуючих конкуруючих продуктів.

Зміст ідеї, напрямки застосування та вигоди для користувача зображено на таб.12.1.

Таблиця 12.1 – опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача

Створення чат-бота для месенджеру Telegram, який налаштовується самим користувачем. Бот інтегрований з існуючою CRM системою користувача	1.Створення чат-боту для CRM-системи, яка заснована на REST архітектурі.	1. Створення чат-боту, який дублює можливості існуючої CRM-системи.
		2. Швидке налаштування та відсутність зайвих витрат.
		3. Швидке оновлення у разі зміни чи оновлення CRM-системи.

12.2 Аналіз потенційних техніко-економічних переваг ідеї

Серед техніко-економічних властивостей ідеї можна зазначити так: інтеграція з існуючою CRM-системою користувача, простота оновлення, зручність для клієнтів користувача, самостійне управління життєвим циклом боту.

Продуктами-конкурентами будемо вважати чат-боти для месенджеру Telegram, можливі для налаштування. Серед продуктів конкурентів можна зазначити такі: reply.ai, Flow XO та Sequel. Результати порівняння зображені на таб.12.2.

Таблиця 12.2 – визначення сильних, слабких та нейтральних характеристик ідеї

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			
		Мій проект	reply.ai	Flow XO	Sequel
1.	Інтеграція з існуючою CRM-системою користувача	Сильна сторона	Слабка сторона	Слабка сторона	Слабка сторона
2.	Графічний інтерфейс для налаштування	Слабка сторона	Сильна сторона	Сильна сторона	Сильна сторона
3.	Простота оновлення	Сильна сторона	Слабка сторона	Слабка сторона	Слабка сторона
4.	Інтеграція зі сторонніми ресурсами	Слабка сторона	Сильна сторона	Сильна сторона	Слабка сторона
5.	Зручність для клієнтів користувача	Сильна сторона	Сильна сторона	Сильна сторона	Сильна сторона

6.	Можливість використання для бізнесу	Сильна сторона	Сильна сторона	Сильна сторона	Слабка сторона
7.	Самостійне управління життєвим циклом боту	Сильна сторона	Слабка сторона	Слабка сторона	Слабка сторона

12.3 Технологічний аудит ідеї продукту

У межах цього підпункту потрібно визначитись з технологіями, на яких буде побудований продукт, який реалізує ідею. Для цього розіб'ємо ідею на складові частини та проаналізуємо наявні технології для їх реалізації і проаналізуємо на таб.12.3.

Таблиця 12.3 – технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
1.	Існування бота для телеграм	Telegram API	Наявні	Доступні
2.	Формат файлу для налаштувань бота	yaml/yml	Наявні	Доступні
3.	Мова програмування	Java	Наявні	Доступні

4.	Робота з файлом налаштувань	Зіставлення значень з файлу налаштувань у об’єктно-орієнтований вигляд	Наявні	Доступні
5.	Робота з url ресурсів користувача	Заповненням шаблону url значеннями об’єктів	Треба розробити	
6.	Робота з повідомленням, яке потрібно показати при переході на пункт меню	Заповненням шаблону повідомлення значеннями об’єктів	Треба розробити	
7.	Робота з відображенням пунктів меню	Пошук в дереві пунктів меню, знаходження поточного пункту меню	Треба розробити	
8.	Середа розгортання	Tomcat за допомогою Spring Boot	Наявні	Доступні
9.	Середа розробки	IntelliJ IDEA	Наявні	Доступні

Обрана технологія реалізації ідеї проекту: Отже програма буде реалізована на мові програмування Java, проект буду розгорнутий за допомогою Spring Boot, для файлу налаштувань було обрано формат yaml чи yml. Програма буду зв’язана з Telegram за допомогою Telegram API. Потрібно буде розробити систему пошуку пунктів меню, заповнення шаблонів для повідомлень та адрес ресурсів існуючої CRM-системи користувача. Програми буде написана у середі розробки IntelliJ IDEA

12.4 Аналіз ринкових можливостей впровадження стартапу

Наступним кроком аналізу буде визначення ринкових можливостей, які необхідно використати в процесі впровадження стартапу, та ринкових загроз, які можуть зашкодити реалізації стартап-проекту. Перш за все потрібно провести аналіз попиту на ринку (таб.12.3).

Таблиця 12.4 – Характеристика потенційного ринку стартап-проекту

№п/п	Показники стану ринку	Характеристика
1.	Кількість головних гравців, од.	6
2.	Динаміка ринку (якісна оцінка)	швидко зростає
3.	Наявність обмежень для входу	Немає
4.	Специфічні вимоги для стандартизації та сертифікації	Реалізація GDPR
5.	Середня норма рентабельності в галузі, %	65%

Можна зробити висновок що наявні всі умови для успішного виходу на ринок.

Далі визначимо потенційну цільову аудиторію продукту (таб.12.5).

Таблиця 12.5 – Характеристика потенційних клієнтів

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Спрощення користування продуктами користувача клієнтам	Телекомунікаційні провайдери	- Покупка підписки на сервіс - Покупка ліцензії на використання сервісу	- Можливість оновлення - Робота з інформацією про клієнтів

Після визначення потенційних цільової аудиторії проведемо аналіз ринку: складемо таблиці факторів, що сприяють ринковому впровадженню продукту, та факторів, що йому перешкоджають (таб.12.6 – таб.12.7).

Таблиця 12.6 – Фактор загроз

№п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Конкуренція	У галузі існують конкуренти, які заробили собі ім’я співробітництвом з відомими компаніями	Чітка маркетингова політика для залучення нових клієнтів.

2.	Поведінковий	Страх використання продукту невідомої компанії	Гнучка система пробного періоду та наголошення акценту на перевагах та відмінностях нового продукту.

Таблиця 12.7 – Фактор можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Високий попит	Більш широке розповсюдження CRM-систем	Можливість для залучення більшої кількості клієнтів
2.	Науково-технічних	Розвиток машинного навчання та аналізу даних	Впровадження підтримки навчання за допомогою штучних нейронних мереж

Наступним кроком буде проведено ступеневий аналіз ринку (таб.12.8).

Таблиця 12.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)

1. Вказати тип конкуренції - олігополія	Існує певне коло конкурентів які хоча і не мають такої вузької спеціалізації проти вже добре себе зарекомендували на ринку.	Постійне вдосконалення та підтримка продукту
2. За рівнем конкурентної боротьби - міжнародна	Компанії конкуренти діють на світовому ринку	Підтримка багатої кількості мов (оскільки повідомлення складає користувач, то ця проблема не є наявною)
3. За галузевою ознакою - внутрішньогалузева	Продукт фокусується на створенні ботів для CRM-систем	Фокус на конкретному напрямку розвитку
4. Конкуренція за видами товарів: - товаро-видова	Конкуренція з іншими програмними продуктами	Розвиток програми для покриття більшої кількості типів використання, аналіз рішень конкурентів
5. За характером конкурентних переваг - неціновa	Інтеграція с CRM-системами	1. Оновлення та покращення продукту 2. Розширення його функціоналу

6. За інтенсивністю - марочна	Наявність унікальної назви	Слідкування за дотриманням авторського права

Після аналізу конкуренції проведемо біль детальних аналіз умов конкуренції в галузі за моделлю Портера (таб.12.9).

Таблиця 12.9 – Аналіз конкуренції в галузі за М.Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Клієнти	Товари-замінники
	reply.ai Flow XO	Розробники програмного забезпечення	Телекомунікаційні компанії	Власні розробки конкретно під існуючу систему
Висновки	Інтенсивна конкуренція з наявними на ринку компаніями	Орієнтовні строки досягнення точки беззбитковості – 5 місяців після завершення розробки	Клієнти диктують зручність продукту	Товари замінники не обмежують роботу на ринку

На основі аналізу конкуренції, а також із урахуванням характеристик ідеї проекту, вимог споживачів до товару та факторів маркетингового середовища визначається та обґрунтовується перелік факторів конкурентоспроможності (таб.12.10).

Таблиця 12.10 – Обґрунтування факторів конкурентоспроможності

№п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1.	Інтеграція з існуючою CRM-системою	Не потрібно створювати щось з нуля, достатньо лише використати існуюче рішення
2.	Просте оновлення під зміни у CRM-системі	Налаштування не займають багато часу. Після оновлення достатньо лише перезапустити сервер.

За визначеними факторами конкурентоспроможності проводиться аналіз сильних та слабких сторін стартап-проекту (таб.12.11).

Таблиця 12.11 – Порівняльний аналіз сильних та слабких сторін продукту

№ п/п	*Фактор конкурентоспроможності*	*reply.ai*						
		–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Зручність налаштування						+	
2	Зручність оновлення		+					
3	Інтеграція з існуючою системою		+					
4.	Ціна					+		

За допомогою SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) визначимо сильні та слабкі сторони продукту, загрози та можливості розвитку (таб.12.12).

Таблиця 12.12 -SWOT-аналіз

Сильні сторони:	Слабкі сторони:

- Працює з існуючою CRM-системою користувача - Простота налаштування - Управління життєвим циклом чат-боту	- Відсутність графічного інтерфейсу для налаштування - Вузька спеціалізація - Відсутня можливість інтеграції зі сторонніми ресурсам
Можливості: - Додавання графічного інтерфейсу	Загрози: - Існує можливість розвинення конкурентів на усунення їх головного недоліку – відсутності інтеграції з існуючою системою користувача

Сформуємо на основі SWOT аналізу альтернативи ринкової поведінки (таб.12.13).

Таблиця 12.13 – Альтернативи ринкового впровадження продукту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Зменшення ціни продукту	менша	такі ж
2.	Незначне збільшення ціни та розширення функціоналу системи	більша	більше

Отже, обрана альтернатива – це незначне збільшення ціни та розширення функціоналу

12.5 Розроблення ринкової стратегії продукту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних

споживачів (таб.12.14).

Таблиця 12.14 – Вибір цільових груп потенційних користувачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Телокомунікаційні компанії з існуючими CRM-системами	Висока	Високий	Низька	Проста
2.	Компанії з існуючими CRM-системами	Висока	Низька	Низька	Складна
Які цільові групи обрано: телекомунікаційні компанії					

Оскільки було вирішено зосередитися лише на одному сегменту ринку, то обрана маркетингова стратегія буде стратегією концентрованого маркетингу.

Визначимо базову стратегію конкурентної поведінки (таб.12.15).

Таблиця 12.15 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопроходьцем» на ринку?	Чи буде компанія шукати нових споживачів,	Чи буде компанія копіювати основні	Стратегія конкурентної поведінки

	або забирати існуючих у конкурентів?	характеристики товару конкурента, і які?	
Ні	Буде шукати нових споживачів та пропонувати рішення користувачам, які мають власний аналог продукту	Буде скопійований підхід графічного налаштування продукту	Стратегія заняття конкурентної ніші

Далі оберемо стратегію позиціювання продукту, основні концепції асоціювання продукту (таб.12.16).

Таблиця 12.16 – Визначення стратегії позиціювання

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1.	Легкість використання продукту	Конкурентна	Підтримка існуючих CRM-систем	Гнучкість, зручність, швидкість

12.6 Розробка маркетингової програми

Розробимо трирівневу маркетингову модель продукту: уточнимо ідею продукту та/або послуги, особливості процесу його надання (таб.12.17).

Таблиця 12.17 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
1. Товар за задумом Система створення чат-боту	Опис базової потреби споживача, яку задовольняє товар (згідно концепції), її основної функціональної вигоди: Спрощення користування продуктами користувача клієнтам
2. Товар у реальному виконанні	Характеристики: - інтеграція з існуючими CRM-системами - швидкість оновлення
3. Товар із підкріпленням	Розширення функціоналу продукту

Далі визначимо рівні цінових меж, якими необхідно керуватись при встановленні ціни на потенційний продукт яке передбачає аналіз ціни на товари-аналоги або товари замінники (таб.12.18).

Таблиця 12.18 – Визначення меж встановлення ціни

№ п/п		Рівень ціни на товари-замінники	Рівень ціни на товари-аналоги	Верхня та нижня межі встановлення ціни на товар
1.		-	20-30 ум.од/міс	300-400 ум.од

Наступним кроком буде визначення оптимальної системи збуту (таб.12.19).

Таблиця 12.19 – Формування системи збуту

№п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1.	Покупка ліцензії на використання продукту.	Постійна доступність на ресурсах виробника	Без посередників	Збут через власні канали

І наостанок розробимо концепцію маркетингових комунікацій, що буде спиратись на вибрану основу для позиціювання(таб.12.20).

Таблиця 12.20 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові групи клієнтів	Ключові позиції, обрані для позиціювання	Завдання рекламного повідомлення	Концепція рекламного звернення
1.	Клієнту будуть обирати зручний та економічно вигідний продукт	Інтернет	Інтеграція з існуючими CRM-системами	Привернути увагу потенційних користувачів	Акцентування на принципових перевагах та зручності продукту

12.7 Висновки

Було проведено аналіз ідеї продукту, техніко-економічних властивостей продукту, проаналізовано сильні та слабкі сторони конкурентів та оцінено принципові відмінності між ними та продуктом, проведено технологічний аудит продукту Також було проведено аналіз ринку, очікувань потенційних

користувачів та як продукт може їх задовільнити, а також стратегію поведінки на ринці. Наостанок була визначена маркетингова стратегія на ринці та канали збуту продукту.

Після проведеного аналізі можна зробити такі висновки.

1) Хоча на ринку достатньо конкурентів продукт має серйозні конкурентні переваги та має можливості усунення недоліків. Також треба зазначити, що були окреслені переваги конкурентів і шляхи їх вирішення. Таким чином продукт може усунити свої слабкі сторони.
2) У розробці продукту можна використовувати доступні, зручні та безкоштовні технології.
3) У продукту є сформована цільова група – це телекомунікаційні компанії. Проте клієнтом може будь яка компанія котра використовує CRM-систему. Таким чином коло потенційних покупців є дуже широким.
4) Продукт має високі шанси бути прибутковим.
5) Подальша імплементація проекту є доцільною.

ВИСНОВКИ

Під час написання магістерської дисертації було розглянуто історію розвитку електронних засобів обміну повідомленням і проаналізовано тенденції розвитку месенджерів та доведено, що проблема розвитку і використання чат-ботів в якості дублера можливостей CRM-системи є актуальною. Було вирішено створити два компонента системи: серверну частину CRM-системи та компонент чат-боту.

Початок роботи над системою почався з розробки функціональних та нефункціональних вимог, що допомогло встановити що саме очікується від системи так як саме вона буде працювати. Причому окремо було проаналізовано що повинен вміти компонент чат-боту та що очікується від чат-боту і серверної частини CRM-системи після їх інтегрування.

Після з'ясування вимог до системи було розглянуто можливі технології, на яких буде створена система. Було прийнято рішення розробляти систему в середі розробці IntelliJ IDEA, яку було обрано за її зручність. Було розглянуті шляхи взаємодії чат-боту і системи Telegram та визначено, що така взаємодія можлива за використання Telegram Bot API. Також було обрано мову програмування Java, яка буде використана разом з Spring Framework, за допомогою якого була узгоджена структура системи, а також спрощено доступ до бази даних. Провайдером бази даних системи було обрано H2 за можливість зберігання даних тільки під час роботи системи та за швидкість його налаштування. Для полегшення маніпуляції з даними було обрано фреймворк Hibernate за допомогою якого пришвидшився процес розробки, оскільки маніпулювати CRUD операціями стало легше. Оскільки проект використовує сторонні бібліотеки, то для управління залежностями було обрано Apache Maven. Також було прийняте рішення тестувати систему за допомогою фреймворку JUnit.

Після формування вимог до системи та технологій, за допомогою яких буде реалізоване рішення було розроблено загальну архітектуру всієї системи,

яка включає в собі створені компоненти чат боту і серверної частини CRM-системи які інтегровані між собою, а також серверу системи Telegram, за допомогою якого чат-бот отримав можливість отримувати повідомлення та надсилати відповідь. Також розглянуто, як саме компоненти та серверної частини CRM-системи мають взаємодії між собою з послідовності з описанням різних рівнів кожного компонента та послідовності їх викликів.

Активний процес розробки було почато з побудови ER-моделі для обох компонентів. Оскільки у розробці було використано Hibernate, то не було необхідності в створення SQL коду для створення таблиць, а лише було потрібно Java класи, які зображали таблиці в базі даних. Завдяки такій реалізації ORM-підходу таблиці були створені автоматично і також були встановлені зв'язки між ними. Такий підхід значно пришвидшив процес розробки, при чому, з урахуванням використання бази даних яка зберігає дані та структуру таблиць лише під час виконання було спрощено оновлення таблиць та розширення функціоналу.

Під час роботи над системою були розглянуті всі бізнес-процеси системи, які відповідають поставленим формальним вимогам. Кожен процес було детально проаналізовано та описано послідовність дій при виконання операцій. Також на прикладі одного з бізнес процесів окремо була проілюстрована послідовність виконання дій та описано як це працює в інтегрованих компонентах системи, що наочно показало що створена архітектура системи повністю відповідає поставленим задачам, а отже її використання є доцільним.

Обидва компонента було створено з використанням шаблону MVC, що дало змогу стандартизувати процес розробки, відокремити такі рівні компонента як: рівень роботи з базою даних, рівень сервісів та рівень контролерів. Завдяки використанню Spring фреймворку було використано принцип програмування на рівні інтерфейсів, а отже створена система стала легкою в розширені функціоналу, а зміни в окремій частині не торкалися інших частин. Також процес зв'язування різних рівнів та впровадження

залежностей було стандартизовано і архітектура компонентів вийшла логічно зрозумілою. Так як Spring реалізує шаблон IoC, то система вийшла слабко зв'язною. Також важливо зазначити, що всі використані бібліотеки є безкоштовними та не займають багато місця. Оскільки кожна з них постійно підтримується та оновлюється, то сама система вийшла оптимізованою, що виражається в загальній швидкодії.

Для конфігурації чат-боту для використання API існуючої серверної частини системи було вирішено використовувати зовнішній файл, який буде містити всі налаштування. Таке рішення дозволило можливість зміни та розширення функціоналу чат-боту завдяки змінам в цьому файлі. Завдяки цьому тепер, в разі зміни функціоналу серверної частини CRM-системи, не потрібно буде вносити зміни в компонент чат-бота та надавати йому додатковий функціонал, а достатньо буде оновити лише файл налаштувань та перезапустити його. Це дозволило набагато пришвидшити процес його оновлення та зробило зручним в використанні. Також правила задання налаштувань було стандартизовано, а отже процес створення налаштувань не потребує багато часу, а також, завдяки використанню формат yaml, самі налаштування мають деревоподібну структуру та є зручними для розуміння.

Так як тестування є важливою частиною процесу розробки, то для початку було створено список тест-сценаріїв які можливо виконати в рамках створеної системи. Всі сценарії було протестовано мануально і порівняно зі список формальних вимог. В результаті була встановлена відповідність очікуваної та актуальної роботи системи. Також, для всіх найбільш значущих частин системи були створені модульні тести, які дозволили переконатись в коректній роботі цих частин. Також треба зазначити, що було проведено не лише позитивне, а й негативне тестування, а отже були розглянути і сценарії, коли поступаючи дані не відповідають очікуваній поведінці системи.

Наостанок було розроблено стратегію розвинення стартап-проекту на основі створеної системи. В рамках цього було проаналізовано сам продукт та його характеристики, його конкуренти, стан ринку та потреби потенційних

клієнтів в такому типу продукту. Також було встановлено стратегію виводу продукту на ринок, цінову політику та канали розповсюдження. Висновком аналізу вийшло те, що продукт має зацікавленість потенційних клієнтів, має конкурентні переваги над схожими продуктами та має потенціал для впровадження.

Підсумовуючи, було проведено повний процес розробки продукту, який включав в себе як і розробку ідеї так її поступову реалізацію та стратегію його комерціалізації. Ключовими відмінностями продукту є те, що він надійний, простий в розширені та універсальним. Продукт є надійним, так як було впроваджено тестування, а також зручним в подальшому розвитку, оскільки створені тести при змінах будуть перевіряти вплив нового функціоналу на старий, причому тести будуть запускатись автоматично. Зручності розширення функціоналу також допомагає можливість додавання нових бізнес-процесів не розробляючи новий програмний код, а отже його розширення не потребує багато коштів та часу, що є дуже вигідним для бізнесу. Завдяки тому, що створений чат-бот є універсальним, то може бути використаним для будь якого типу та розміру компаній та сфер їх використання, оскільки єдиною вимогою є наявність власної CRM-системи можливості якої він буде використовувати. Результатом аналізу стартап-проекту було встановлення того, що продукт може зайняти свою нішу на ринці та має потенціал для розвитку та покращення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ICQ [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/ICQ
2. Talcomatic [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Talkomatic
3. PLATO [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/PLATO_(computer_system)#Online_community
4. 15 Amazing Telegram Stats and Facts(2019) [Електронний ресурс] – Режим доступу до ресурсу: https://expandedramblings.com/index.php/telegram-stats
5. Most popular global mobile messenger apps as of October 2019, based on number of monthly active users [Електронний ресурс] – Режим доступу https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps
6. Market reach of the most popular mobile app categories in the United States as of September 2019 [Електронний ресурс] – Режим доступу: https://www.statista.com/statistics/579302/top-app-categories-usa-reach
7. ПОЗИТИВНОЕ И НЕГАТИВНОЕ ТЕСТИРОВАНИЕ [Електронний ресурс] – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical-articles/positive-negative-testing/
8. Unit-тесты [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/116372/
9. Mockito и как его готовить [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/444982/
10. Webhook [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Webhook

11. Bots: An introduction for developers [Електронний ресурс] – Режим доступу до ресурсу: https://core.telegram.org/bots
12. Representional state transfer [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Representational_state_transfer
13. Application programming interface [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Application_programming_interface
14. Spring Framework Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/docs
15. Maven – introduction [Електронний ресурс] – Режим доступу до ресурсу: https://maven.apache.org/what-is-maven.html
16. JUnit 5 User Guide [Електронний ресурс] – Режим доступу до ресурсу: https://junit.org/junit5/docs/current/user-guide/
17. Unit tests with Mockito – Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.vogella.com/tutorials/Mockito/article.html
18. Spring in Action 4th edition / C.Walls. – Manning Publictions, 2014. – 624с.
19. Визначення засобів розробки чат-бота «помічник абітурієнта» для сучасних месенджерів [Електронний ресурс] / В.Скороход – Режим доступу до ресурсу: https://phm.kspu.kr.ua/nauka/konferentsii/fizyka-tekhnolohii-navchannia/99-2017/kompiuterni-nauky-ta-informatsiini-tekhnolohii/1118-vyznachennya-zasobiv-rozrobky-chatbota-pomichnyk-abituriyenta-dlya-suchasnykh-mesendzheriv.html
20. Head First Design Patterns / E.Freeman, K.Sierra, B.Bates, E.Robson. - O’Reilly Media, 2004 – 638с.